DEVELOPMENT OF FREE/LIBRE AND OPEN SOURCE SPATIAL DATA ANALYSIS SYSTEM FULLY COUPLED WITH GEOGRAPHIC INFORMATION SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

VOLKAN OSMAN KEPOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
GEODETIC AND GEOGRAPHIC INFORMATION TECHNOLOGIES

MARCH 2011

Approval of the thesis:

DEVELOPMENT OF FREE/LIBRE AND OPEN SOURCE SPATIAL DATA ANALYSIS SYSTEM FULLY COUPLED WITH GEOGRAPHIC INFORMATION SYSTEM

Submitted by **VOLKAN OSMAN KEPOĞLU** in partial fulfilment of the requirements for the degree of **Doctor of Philosophy in Geodetic and Geographic Information Technologies Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Vedat Toprak _____________________
Head of Department, **Geodetic and Geographic Information Technologies**

Prof. Dr. Şebnem Düzgün _____________________
Supervisor, **Mining Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Oğuz Işık _____________________
City and Regional Planning Dept., METU

Prof. Dr. Şebnem Düzgün _____________________
Mining Engineering Dept., METU

Assoc. Prof. Dr. Ahmet Coşar _____________________
Computer Engineering Dept., METU

Assoc. Prof. Dr. Nurunnisa Usul _____________________
Civil Engineering Dept., METU

Assoc. Prof. Dr. Çiğdem Varol _____________________
City and Regional Planning Dept., Gazi University

Date: _____31 / 03 / 2011_____

ABSTRACT

DEVELOPMENT OF FREE/LIBRE AND OPEN SOURCE SPATIAL DATA ANALYSIS SYSTEM FULLY COUPLED WITH GEOGRAPHIC INFORMATION SYSTEM

Kepoğlu, Volkan Osman
Ph.D., Department of Geodetic and Geographic Information Technologies
Supervisor: Prof. Dr. Şebnem Düzgün

March 2011, 220 pages

Spatial Data Analysis (SDA) is relatively narrower and constitutes one of the areas of Spatial Analysis. Geographic Information System (GIS) offers a potentially valuable platform for supporting SDA techniques. Integration of SDA with GIS helps SDA to benefit from the data input, storage, retrieval, data manipulation and display capabilities of GIS. Also, GIS can benefit from SDA techniques in which the integration of these techniques can increase the analysis capabilities of GIS. This integration serves for disseminating and facilitating improved understanding of spatial phenomena.

How SDA techniques should be integrated with GIS arise the coupling problem. The complete integration of SDA techniques in GIS can be applied without the support of GIS vendor when the free/libre and open source software (FLOSS) development methodology is properly followed. This approach causes to interpret the coupling problem in a new way. This thesis aims to develop a fully coupled SDA with GIS in FLOSS environment. A fully coupled SDA in free GIS software as FLOSS system is developed by writing nearly 13,000 line Python code in 2.5 years. Usage of this system has reached to nearly 1600 unique visitors, 3000 visits and 8600 page views in two years.

As the current status of development in GIS is considered, it is unlikely in commercial market to have full coupled SDA techniques in GIS software. However, it is expected to have more SDA developments in proprietary GIS software in the near future as there is an increasing trend for requesting more sophisticated SDA tools.

Keywords: GIS, Spatial Data Analysis, Free Software, Open Source Development

ÖZ

COĞRAFİ BİLGİ SİSTEMİ İLE ENTEGRE ÖZGÜR VE AÇIK KAYNAK KODLU MEKANSAL VERİ ANALİZ SİSTEMİNİN GELİŞTİRİLMESİ

Kepoğlu, Volkan Osman
Doktora Jeodezi ve Coğrafi Bilgi Teknolojileri Bölümü
Tez Yöneticisi: Prof. Dr. Şebnem Düzgün

Mart 2011, 220 sayfa

Mekansal Veri Analizi (MVA) mekansal analiz alanına göre daha dar bir alandır. Coğrafi Bilgi Sistemi (CBS) MVA tekniklerini işlemek için potansiyeli yüksek ve değerli bir platformdur. MVA teknikleri CBS ile tamamen entegre edildiğinde, MVA, CBS'nin veri girişi, depolama, geri getirme, veri işleme ve görüntüleme yeteneklerinden yararlanabilecektir. Ayrıca, MVA tekniklerinin entegrasyonu sayesinde CBS kendi analiz yeteneklerini daha da artırabilir. Bu entegrasyon, mekansal olayların daha iyi anlaşılarak yaygınlaştırılmasına ve kolaylaştırılmasına hizmet edecektir.

MVA tekniklerinin CBS ile nasıl entegre edileceği bağlantı probleminin konusudur. Özgür ve açık kaynak kodlu yazılım (ÖAKKY) geliştirme metodolojisi uygun bir şekilde gerçekleştirilebilirse ticari bir CBS yazılım firmasının desteği olmadan CBS ile MVA teknikleri tümüyle entegre edilebilir. Bu yaklaşım bağlantı probleminin yeni bir şekilde yorumlanmasına neden olmaktadır. Bu tezde, MVA ile CBS'nin ÖAKKY ortamında tamamen entegrasyonun sağlanması amaçlanmıştır. 2,5 yıl içinde MVA teknikleri ile entegre özgür bir GIS yazılımının ÖAKKY sistemi olarak geliştirilmesi için yaklaşık 13.000 satır Python kodu yazılmıştır. Geliştirilen bu sistemin kullanımı iki yıl içerisinde yaklaşık 1600 tekil ziyaretçi, 3000 ziyaret ve 8600 web sayfası izlenim sayısına ulaşmıştır. Bu rakamlar MVA tekniklerinin kullanıcı dostu CBS yazılımı ile entegre edilmesine duyulan bir ihtiyacın kanıtıdır.

CBS alanındaki mevcut durumun gelişimi dikkate alındığında, tüm MVA tekniklerinin CBS yazılımları ile entegre edilmesinin ticari pazarda mümkün olmadığı gözükmektedir. Ancak, MVA için daha gelişmiş araçlara artan eğilim nedeniyle yakın gelecekte daha fazla MVA geliştirmelerinin sahipli CBS yazılımlarında da gerçekleşebileceği düşünülmektedir.

Anahtar Kelimeler: CBS, Mekansal Veri Analizi, Özgür Yazılım, Açık Kaynak Kodlu Yazılım Geliştirme

ACKNOWLEDGEMENTS

I would hereby like to express my sincere appreciations to my supervisor Prof. Dr. Şebnem Düzgün for her great guidance and support. This development could not be completed in the FLOSS environment without her encouragement and motivation.

I would like to thank to Chair of my committee, Prof. Dr. Oğuz Işık and committee members; Assoc. Prof. Dr. Ahmet Coşar, Assoc. Prof. Dr. Nurunnisa Usul and Assoc. Prof. Dr. Çiğdem Varol for their guidance, critics, motivations and invaluable discussions.

I would also like to thank all members of QGIS community, especially to Gary Sherman for setting up user friendly environment of QGIS, to Barry Rowlingson, Martin Dobias, Carson Farmer, Anne Ghisla, Borys Jurgiel, Horst Düster, Adrian Baddeley, Roger Bivand, Giovanni Manghi and all my users for testing the system, using the plug-in, reporting bugs and motivating me to add more tools.

I would like to add special thanks Ulaş Alagöz for his help, Servet Ahmet Çizmeli for setting access to Debian Server, Burak Beyhan for motivation in the FLOSS subject and Osman Yalçın for valuable contribution to FLOSS.

Finally, I would like to thank to my family especially my father, Günaydın Kepoğlu for his strong patience, my wife, Arzu Kepoğlu for her great support and my daughter, Nil Kepoğlu for her understanding and love. Without their encouragements and motivations, this thesis could not be completed.

To My Dear Wife and Lovely Daughter

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

API	Application Programmers Interface
ASCII	American Standard Code for Information Interchange
CRAN	Comprehensive R Archive Network
ESDA	Exploratory Spatial Data Analysis
ESRI	Environmental Systems Research Institute
FLOS	Free/Libre and Open Source
FLOSS	Free/Libre and Open Source Software
GDAL	Geospatial Data Abstraction Library
GIS	Geographic Information System
GNU	Recursive Acronym for GNU is Not UNIX
GPL	General Public Licence
GRASS	Geographic Resources Analysis Support System
GUI	Graphical User Interface
OS	Operating System
POSIX	Portable Operating System Interface for UNIX
QGIS	Quantum GIS
RS	Remote Sensing
SDA	Spatial Data Analysis
SDA4PP	Spatial Data Analysis for Point Pattern
XML	Extendable Markup Language

CHAPTER 1

INTRODUCTION

1.1. Introduction

Spatial analysis is the quantitative study of phenomena located in the space. Spatial Data Analysis (SDA) is relatively narrower and constitutes one of the main areas of spatial analysis. SDA is the combination of statistical description and modeling of spatial data. SDA requires very tedious computation processes and Geographic Information System (GIS) offers a potentially valuable platform for processing SDA techniques. SDA should benefit from the data input, storage, retrieval, data manipulation and display capabilities of GIS. Also, GIS can benefit from SDA techniques in which, the integration of these techniques can increase the capabilities of GIS in exploring, analyzing and predicting the behavior of the spatial processes. This integration serves for disseminating and facilitating the SDA field.

The complete integration of SDA in GIS approach is eliminated from the initial stage of GIS software development process. Most of the SDA techniques were developed when the GIS is at the initial stage. In 1960s, the urgent need of GIS was to perform the geometry related operations such as overlay, point-in-polygon, intersection etc. At that time, the computation power of computer was very limited. Since 1980, GIS vendor’s main motivations have become in the utilities and local government, rather than in the SDA research community (Fotheringham and Rogerson 1995). These areas are more profitable for GIS vendors than the scientific research (Bailey and Gatrell 1995). These are historical reasons that have avoided integrating SDA techniques in GIS.

GIS users gradually have started to request more SDA techniques in GIS software. However, the number of SDA techniques in the GIS software is not adequate especially when the number is compared with the available SDA techniques coming from different application areas such as geography, biology, epidemiology, statistics, geographic information science, remote sensing, computer science, mathematics, and scientific modeling. For example, although Environmental Systems Research Institute (ESRI), a commercial GIS company, is the world leader in GIS and developing GIS software since 1969, ArcGIS, a group of famous GIS software products, has no SDA tools in version 8.0 released at the end of 1999. Two extensions, namely; spatial and geostatistical analyst, are available which are both related with the field of spatial analysis. SDA tools are added in a new toolbar, named as spatial statistics analysis, with the release of version 9.0 in May 2004. The number of available SDA tools in version 9.1 is 10. This figure has increased to 14 with the version 10.0, released in June 2010. However, available SDA techniques over total number of all tools in ESRI ArcInfo licensed ArcGIS Desktop Version 10.0 is only 1.01% (14/1387). Until 2004, there was no SDA tool in ArcGIS Desktop, but after 2004, the number of available SDA tools is gradually increasing, although the progress of adding new SDA tools is slow and not adequate in number. SDA techniques were not known among GIS users because of not having enough SDA tools in GIS software. In order to disseminate and facilitate the SDA field, there should be a system that many SDA tools should be available in GIS software. The request of having SDA tools in the GIS software is continuously increasing with the growth of GIS at the last decade.

Spatial analysis tools such as point in polygon, buffer, slope, intersection, union, etc. can be used easily because they are available in GIS software; however, a few SDA techniques have been implemented in the GIS Software. The demand of requesting more SDA tools for doing spatial research in the GIS software has gradually increased as geospatial industry grows. Several SDA tools can be found either in GIS software or in statistical package in the form of a script file, add-on, package, extension, toolbar, module, or development library. SDA techniques are scattered all over these applications. Some of them are in the specialized niche SDA programs that have limited in number of SDA techniques and GIS functionality.

Quadrat Based Methods, Kernel Estimation, Adaptive Density, Nearest Neighbor Distance, Estimate Empty Space, Ripley's K and L Functions and their local functions, Clustering, Space-Time Clustering, Variance/mean Ratio Tests for Quadrat Counts and Tests, Simulation Envelope of Complete Spatial Randomness, Poisson Model and Spatial Kolmogorov and Smirnov Test are some of the examples for SDA techniques. These techniques are not applicable as spatial analysis tool in the GIS software, because these tools could not be found in the generic GIS software. The usage of these functions as compared with buffer tool is not common and also many GIS users do not know that there is such an analysis technique. These are the practical difficulties to apply the SDA techniques, because they do not exist in the GIS software. SDA techniques can be used widely among GIS users when SDA techniques become available in GIS software.

Many GIS users become accustomed to use menu driven GIS software, so that, the GIS users want SDA techniques to apply in the user friendly, visual and interactive software. However, the cost of developing such a software product could not be recovered with the selling of license fee, because the request is not strong enough to support the new development expenses of standalone SDA software. The recent and famous development example for creating such software in exploratory SDA field is GeoDa. The development cost of GeoDa is financed by a grant provided by US National Science Foundation. The amount of grant was $4,335,573 (NSF 2007). The grant was initialized in 1999 and finalized in September, 2004. The development of GeoDa is ongoing in order to reach more mature and stable architecture in August, 2010.

Haining (1989) states that specialized SDA software development has been demanded since the late 1980s. This necessity has come from GIS researchers. GIS researchers have requested more available tools to perform spatial analysis in a GIS environment in order to increase their understanding of spatial processes. The need for computer software to facilitate SDA field has also been mentioned by Anselin (1992), Goodchild et al. (1992), Fotheringham and Rogerson (1993), and Anselin et al. (2006). Integrating SDA techniques with GIS environment is discussed conceptually by these SDA theorists. It can be concluded from these discussions that an easy to use, visual and interactive software package is required in order to disseminate and facilitate the SDA field.

Fotheringham and Rogerson (1993) explain that spatial (data) analysis is less rapid developed area then GIS. The lack of integrating SDA with GIS has hindered the spread of GIS, but the progress of integration is inevitable due to demand for increasing the analytical capabilities of GIS. Fotheringham and Rogerson (1993) mention that when SDA techniques are integrated within GIS, future users of GIS may misuse these techniques due to major problems of SDA like “modifiable areal unit problem, boundary problems, spatial interpolation, spatial sampling procedures, spatial autocorrelation, goodness-of-fit in spatial modeling, context-dependent results and nonstationarity, and aggregate versus disaggregate models”. They hope that GIS may circumvent these problems by helping the users to learn these types of fundamental problems in SDA. It is worried that while using SDA techniques in the simplistic environment of GIS, these problems will be ignored and the result of the analysis will be interpreted poorly and in an inaccurate way. Fotheringham and Rogerson (1993) give an example for the similar situation which occurred in the statistical software packages. Misuse of statistical methods are facilitated in these packages, however, they also helped to disseminate the statistical methods that otherwise would not have been applied.

1.1.1. Coupling of SDA and GIS

In 1990s, SDA theorists discussed the integration of SDA and GIS widely. Goodchild et al. (1992) conceptualized the coupling problem into four strategies:

1. *Loose coupling of proprietary GIS software with statistical software*
2. *Close coupling of GIS and statistical software*
3. *Standalone spatial analysis software*
4. *Complete integration of statistical spatial analysis in GIS*

First coupling strategy is to exchange the data between GIS software and statistical package. This exchange can be done based on ASCII or common spatial binary format. The spatial data in GIS software is converted as input into statistical package. The analysis is done in the statistical package. Mostly, this kind of analysis in the statistical package causes spatial autocorrelation problem. And then, the output of package is converted as an input for GIS software. Although this conversion provides a solution, it is impractical and inefficient.

The conversion between GIS software and statistical package can be done at the background without inferring the user when the statistical analysis routines are embedded in the GIS software. This is known as second coupling strategy. SDA theorists demonstrated some close coupling software examples. SpaceStat and DynESDA are two extension examples developed for ESRI ArcView GIS series 3.

These examples are developed by SDA theorists, not by GIS vendors. These close coupling examples do not convince the GIS vendors to add extensions in the main source code of their program, because the integration of extension will increase the complexity of program and they are not profitable for vendors to spend effort to add these tools that will be rarely used by the users. When the architecture of GIS software is changed radically by ESRI, the close coupling strategy has become obsolete and the development effort could not be continued, because the extension could not be upgraded to new version of GIS software. The final version of ArcView GIS was version 3.0a in 1997 and the last release of this software was version 3.3 in 2002. Instead of developing ArcView GIS series 3, ESRI has decided to develop new, visual and easy to use GIS product, named as ArcGIS, which is the integration of window based use of ArcView GIS series 3 and command line of ArcInfo workstation. ArcGIS Desktop version 8.0 is released in late 1999. The architecture of ArcGIS and ArcView GIS is totally different which caused that the SDA extensions could not work with next new version of GIS software.

Third coupling strategy is the development of standalone spatial analysis software. Info-Map is one of standalone SDA software examples. It is distributed with a book, written by Bailey and Gatrell, and published in 1995. Info-Map is a remarkable example which runs only in the Microsoft DOS operating system. However, the software can perform the following SDA techniques; kernel estimation, K functions, variogram estimation, kriging, spatial autocorrelation analysis, spatial regression, principal components analysis, multivariate classification and non-spatial regression. It is surprising that all these tools are available in 1995 and working only in the Microsoft DOS operating system. However, today, Microsoft DOS operating system does not exist anymore and nobody can use this operating system properly. Info-Map software cannot be used now, because the originators did not upgrade the architecture of the software in parallel with the changes in the operating system.

Another standalone SDA software product is SpaceStat, which is developed by Anselin with the support of a number of grants from the US National Science Foundation between 1983 and 1994. Anselin (1992) explains that SpaceStat is first released by the US National Center for Geographic Information and Analysis in 1992. SpaceStat is updated to version 1.8 and 1.9 in 1995 and 2000, respectively. SpaceStat is purely a 32 bit DOS product, which runs in Windows 95, 98 and NT.

SpaceStat version 1.91 has a command line interface like Microsoft DOS operating system in 2002. SpaceStat has also become out-dated like Info-Map.

Another example for the third coupling strategy is GeoDa software. This software has been developing by Anselin since 1999. Anselin et al. (2006) explain that GeoDa is developed in order to provide a needed tool for disseminating and facilitating SDA field. This package should be easy-to-use, visual, and interactive software for GIS users. The cost of developing GeoDa software is financed by a grant provided by US National Science Foundation. The first official release of GeoDa version 0.9 was in February, 2003. The bug free or stable version of the GeoDa was announced in September, 2004. The development methodology of this software is for the closed source environment. This version can work only on Microsoft Windows operating system version XP. In every 2.18 years, Microsoft develops a new version of operating system which means that radical change may occur in the structure of operating system within this period.

Like Info-Map and SpaceStat, GeoDa also become out-dated software. The problem was related with the MapObjects-Java library which was not included deliberately by Microsoft in the Windows Vista. Not only GeoDa but also ESRI's products rely on the MapObjects-Java libraries. The problem was so severe that even the power of ESRI was not enough to solve the problem of not running ESRI ArcGIS Desktop 9.2 in Windows Vista in May 2007. ESRI could fix the problem with the next release of their products. ESRI ArcGIS products have supported Windows Vista with the release of version 9.3 in June 2008. The sustainability of GeoDa has broken with Windows Vista in March 2007 after two and a half years from the release of stable version. The cost of adapting such a modification is insuperable when especially the support of fund is cut and there is no support from any commercial vendor or community. The cost of keeping the software functional could be unmanageable without the support of any vendor in every 2.18 years in Windows operating system. This phenomenon, becoming out-dated, is very common for SDA software like Info-Map, SpaceStat and GeoDa that is designed to work only in Microsoft environment. Much SDA software working only in closed source environment has become quite out-dated. Under a closed source software development methodology, the decision of either upgrading or updating the architecture of the software belongs to only the originator/owner of the software, which is one of the constraints for the sustainability of standalone SDA software. Although the development of GeoDa has created valuable contributions to the SDA community and made many of the SDA researchers' works easier, the software has become quite out-dated in a short time due to applying misleading software development methodology in 2000s as well as applied in 1990s.

Fourth type of coupling problem is the complete integration of SDA in GIS which is unrealistic from the GIS vendors' point of view for the time being. The main reason is that there is a weak demand from the market for the use of SDA techniques in GIS. The integration request has only come from the scientific research community. This community is not big enough to support the new developments. For example, GeoDa is one of the most famous world-wide, user friendly, closed source, freeware and standalone SDA software. The number of GeoDa users grows over 52,000 in 7.5 years between February 2003 and August 2010. This number is not huge enough to be a market for the GIS vendors. That's why; applying this integration by the GIS vendors has not become profitable.

1.1.2. Need for FLOSS Development

A software environment is required for SDA field which have many tools performing SDA techniques in GIS software. This software environment should be provided at no cost or very low fee in order to reach more GIS users. In this environment, GIS users can perform both GIS and SDA tools in user friendly and menu driven system. Also, SDA researchers should have a right to access the source code of each SDA technique in order to examine and understand the analysis techniques and develop new ones. This software development environment should be open even to external developers. SDA field can become more known and widely used application area among GIS users when SDA techniques are available to use in user friendly GIS software.

Accessing the source code of technique is vital for SDA field in order to understand exactly how the SDA technique is applied. This characteristic property is valid for SDA field, because many application areas have contributed to the development of SDA field. Many areas have caused to develop different style of analysis. Bailey and Gatrell (1995) explain the meaning of this variety as that some of existing SDA techniques are re-invented. The subject of SDA field is scattered in different fields. Newcomers think that this field is made up from different techniques without basic any theory. This historical development is one of the drawbacks of SDA which complicates the techniques to be understood easily and used widely. Therefore, accessing the source code of SDA technique is a predominant factor in the development of specialized GIS software for dissemination and facilitation of SDA field.

Researchers, SDA theorists, GIS experts and developers can examine the SDA tool, improve the algorithms, develop new techniques and eliminate the possible errors and bugs, if they can access the source code of the tool. This group can also enhance the GIS software by integrating new SDA techniques. In order to provide such an opportunity, accessing the source code of the software is the compulsory starting point. The opportunity of accessing and modifying the source code of the tools and redistributing the derived source code is naturally provided as a right for everyone only in the Free/Libre and Open Source Software (FLOSS) development environment. This right is granted legally by free and open source licenses. There are hundreds of FLOSS licenses. Each type of license gives different rights for their users. For example, the license of Python programming language gives the right to use the language for free also for commercial purposes. If the development is done in Free/Libre and Open Source (FLOS) GIS Software, GIS users can use this product without paying any license fee for each copy of the software. This is the cost advantage of FLOSS environment, which could accelerate the dissemination and facilitation of SDA field.

Dissemination and integration of SDA techniques with GIS arises the coupling problem. The full coupling problem of SDA in GIS has not been fully solved, because the development of SDA techniques in user friendly GIS software has not been achieved yet due to the lack of support from the commercial market. The support of any vendor is not the essential factor. The complete integration of SDA techniques in GIS can be applied without the support of any commercial GIS vendor when FLOSS development methodology is properly followed. In this thesis, SDA System refers to integration of SDA techniques in GIS software in the FLOSS environment. The development of SDA system in other words integrating SDA techniques in GIS software in FLOSS environment is the new interpretation of coupling problem. This is the most appropriate way in order to disseminate and facilitate the SDA field. The sustainability of SDA system could be guaranteed in FLOSS environment, if the support of researchers, SDA theorists, GIS experts and developers could be made up as a community around the FLOSS project. Binding FLOS spatial analysis libraries with FLOS GIS software have tremendous potential for disseminating and facilitating SDA field. This kind of development strategy can create user friendly and FLOS SDA system for GIS users, developers and researchers.

In this thesis, it is significant to know the distinction between freeware and free software. The distinction between these two terms can sometimes be misused. Freeware is software that is available for use at no cost, whereas free software can be used, copied, studied, modified, distributed and redistributed in modified or unmodified without any restriction both with in the form of binary and source code. Sole freedom provided by freeware is to use the software for free. The source code of freeware software is never distributed to the users. For example; Anselin et al. (2006) define that GeoDa is a free software program. This definition is misleading, because GeoDa is freeware software, but not free software. This difference creates to follow up different software development methodology. This thesis emphasizes this distinction in favor of free software due to available freedoms in this environment.

1.2. Motivation and Objective of the Thesis

In this thesis, it is aimed at developing a fully coupled SDA with GIS in FLOSS environment. The integration of SDA techniques in GIS can be done with FLOSS tools without the support of any vendor, which causes to interpret the coupling problem in a new way. As the current status of GIS is considered, it is unlikely in the commercial market to have full coupled SDA techniques in GIS software. However, it is expected to have such integrations in near future as there is an increasing trend for more sophisticated tools for spatial analyses. The development carried out in this thesis has showed that SDA techniques should be completely integrated in GIS with applying FLOSS development methodology.

This thesis focuses on the development of SDA system where many SDA techniques are applied in FLOS GIS software in FLOSS environment. This statement has been demonstrated by developing SDA System with FLOS tools. The development of FLOS SDA System has been done by writing approximately 13,000 lines as Python code (12835 lines in the latest revision 0.196 in 2.5 years) between

January 2008 and June 2010. The developed system communicates with FLOS statistical package and GIS Software. Source code of the system is open to use for everyone in any time, since the source code can be accessed in the repository through the Internet since June 2008. This openness is provided by publishing the source code in the repository under the web site of the Department of Geodesy and Geography Information Technologies in Middle East Technical University in order to apply FLOSS development methodology properly. The outcome of the system is used and mostly tested by QGIS community. The progress of the development with screenshots of implemented tools and the revision history can be followed at the web site of the developed FLOS SDA system under the following web address; http://ggit.metu.edu.tr/~volkan/index.html.

The amount of interest of SDA system is related with the size of QGIS community, because the development has used QGIS community resources. It is difficult to guess the size of QGIS community; however, one statistical figure can be given in order to give an idea for the size of community. Windows standalone installer of QGIS version 1.4 is downloaded 71,433 times between January and August 2010. This download number has reached to 96,791 for QGIS version 1.5 in 4 months after the release date in 01.08.2010. This figure shows that how the QGIS community is growing in time. Nearly 100,000 QGIS user is quite fascinating for FLOS desktop GIS software product. It is thought that the size of SDA market is much smaller than this figure. The number of GeoDa users can be accepted as the sign for determining the size of SDA users, because GeoDa is the most famous world-wide used, user friendly, closed source, freeware and standalone SDA software. The number of GeoDa users grew over 52,000 in 7.5 years between February 2003 and August 2010. The number is not large enough to be a market for the GIS vendors. That's why; the integration of SDA in GIS has not become profitable for the GIS vendors.

Alternative of developing FLOS SDA system is the development carried out in the closed source environment. This environment is not sustainable for standalone SDA software due to weak support of SDA market. For example, if the coupling of SDA techniques in GIS had been done in proprietary GIS product like ESRI's desktop GIS software; ArcGIS Desktop, the outcome of the development would be an extension, which could work only in Microsoft Windows operating system and the development of extension could be naturally designed for closed source environment. In this environment, the sustainability of the development does belong to either GIS vendor or Microsoft Company, but not to the developer of the extension. The originator's responsibility is only to follow their architectural changes which will occur in approximately every two years. This adaptation should be carried out as maintenance work in the closed source environment in order to keep the software up-to-date. If this maintenance effort has not been carried out, the extension will become out-dated and have antiquated architecture in time like what happened in several SDA extension and standalone SDA software. Keeping up-to-date the software is not an easy task for the developers, especially where there is a lack of market support and in every two years the architectural schema of operating system changes dramatically. The same situation becoming out of date has been seen in several SDA applications like Info-Map, SpaceStat Extension, DynESDA Extension and GeoDa. These SDA applications are designed to run only in Microsoft Windows operating system. All of them have become out-dated with their antiquated architecture. SDA market is not strong enough to compensate the upgrade cost of these software architectures.

Another reason of becoming out-dated is that all these software are designed to develop in the closed source environment. When the originator has decided not to develop the software anymore, it is not possible to continue to develop and maintain the SDA software. At first, technically, it is not practical, because source code of the software is closed, and secondly, legally, continuing to develop the software by other developers requires the permission of the originator. Without the permission and the use of source code, it is forbidden and impractical to upgrade and develop the SDA software.

Info-Map, SpaceStat Extension, DynESDA Extension and GeoDa are designed to work only in one operating system. When a dependent library has not included or changed either in the operating system or in the GIS software, the development would face the risk of becoming out-dated. Who would carry out the further maintenance work and upgrade the architecture of the software is not clear in the closed environment, when the originator of the software has decided to stop developing the software. The originator of the software is only the person who decides to continue to develop the software in the close source environment. Due to lack of market for SDA field, a different software development methodology and environment is required in order to carry out maintenance work, to provide continuity of software and to upgrade the architecture of the software. In order to provide more sustainable environment for SDA software, there are several requirements:

- When the originator of the software has decided to stop developing the software, other developers can continue to develop, upgrade and maintain the software without the permission of the originator. This opportunity should be sustained legally with licenses.
- Source code of the software should be accessible from the Internet, and can be changed and distributed via Internet.
- Software development process should be open to every developer and they can follow the progress of the development.
- Each developer can participate to take the strategic decisions about the development of the software.
- Software should be cross-platform running in several operating systems like Windows, Linux and Mac OS.

These requirements are met in the Free/Libre and Open Source Software (FLOSS) environment. FLOSS licenses especially GNU (a recursive acronym for GNU is Not UNIX) General Public License (GPL) provides these requirements as a right for every user. In the GNU GPL, there are four types of freedom:

1. *The freedom to run the program, for any purpose,*
2. *The freedom to study how the program works,*
3. *The freedom to redistribute copies so you can help your neighbor,*
4. *The freedom to distribute copies of your modified versions to others.*

All these freedoms match with the SDA software requirements. This sustainability is the first motivation factor for this thesis to make the coupling in the FLOSS environment.

The second motivation factor is the commercialization of FLOSS environment. In the last decade, Linux has showed a rapid progress, reached to many users from different countries and in each day Linux support has grown. The use of Linux has spread to worldwide and the number of Linux distributions formed by different organizations has exceeded couple of hundreds. Red Hat, IBM, Novell, Intel, Google and Oracle have supported the FLOSS development methodology. ESRI has changed the script languages of ArcGIS Desktop from Arc Macro Language to Python programming language since May 2004 due to benefits of FLOS and the capabilities of Python programming language.

A commercial business can find high quality and specialized applications in various subjects in the FLOSS world. Today FLOSS reached such an enormous variety that all daily used products and tools such as operation system, office application, web browser, e-mail application, database application, programming language, editor, image, music and graphic tools are offered by FLOSS world. For example, Mozilla Firefox (web browser), Mozilla Thunderbird (e-mail application), OpenOffice (free alternative to Microsoft Office), Ubuntu (operating system; one of Linux distribution), MySQL (database application), PHP (programming language) and Apache (HTTP server) are several examples for the well known FLOS applications. Firefox is the reinvention of the Netscape by Mozilla Company. Owner of the MySQL and OpenOffice is Sun Microsystems which is bought by the Oracle Company. Ubuntu is sponsored by private company whose owner is multi-millionaire Mark Shuttleworth. In opposition to these projects, PHP and Apache projects have initiated by several enthusiast developers, not with the support of commercial corporate companies. They succeeded to keep going continuous development ever since the beginning of the project to today, because their product has created a valuable thing and filled the gap better in quality and performance than other products could do. For example, Netcraft (2010) states in July 2010 Web Server Survey that Apache has been the most popular HTTP server in use since 1996 and served more than half of all the websites. There are some FLOS projects like Apache that the project is supported by active community all around the world, which makes them the most well known and used programs in their application areas.

FLOSS development methodology is started to be used also as a business model. This model is based on mainly providing technical service rather than making profit by selling license fee. Many commercial software developer companies have adopted this method with parallel traditional software development method, because this method decreases the cost of software development, allows widest distribution of software with lost cost and provides more user oriented approach.

The third motivation of this thesis is the development date of GeoDa. Anselin explains in 2006 that the outcome of effort for the development of GeoDa is to create visual, interactive and user friendly standalone SDA software. Therefore, the development of SDA system is still valid in 2000s as well as in 1990s. In addition to these, the development of GeoDa is still continuing in August 2010 in order to have cross platform support and to reach mature state with the stable release of GeoDa.

1.3. Organization of the Thesis

First chapter introduces the subject briefly and continues to explain the motivation and objective of the thesis.

Literature review is composed of Chapter II and III. Chapter II reviews a brief overview of SDA field. Review is also associated with coupling problem in this chapter. Last part of this chapter explores several SDA extensions and standalone applications, and their executable status.

Chapter III defines the term “free software” and “open source”. FLOSS development and its benefits are discussed in the following sections. This chapter ends with the examples of FLOSS GIS libraries and applications.

Methodology of the thesis is explained in Chapter IV. This chapter also addresses the properties of the components of SDA system.

Implementation part for the development of FLOS SDA system is explained in Chapter V in which the outcome of the system, SDA4PP Plug-in, is explained based on the example of the source code of one of implemented SDA tools.

In the last chapter, the statistics of SDA system’s users are examined in detail as a sign for the success of the development of FLOS SDA system. This chapter ends with a discussion and conclusion part as well as recommendations provided for further studies.

CHAPTER 2

SPATIAL DATA ANALYSIS (SDA)

2.1. SDA versus Spatial Analysis

Spatial Data Analysis (SDA) is one of the applied spatial research fields, which is different from geostatistics, artificial intelligence, network analysis, computational geometry, spatial econometrics and space time fields. Goodchild (1987 cited in Goodchild et al. 1992) refers to SDA as “a set of techniques devised to support spatial perspective on data”. Goodchild et al. (1992) emphasize the distinction of this field that SDA techniques “are dependent on the locations of the objects or events being analyzed”. Therefore, SDA techniques should require accessing to “both the locations and the attributes of objects”. Bailey and Gatrell (1995) explain that the objective of SDA is to increase basic understanding of the spatial process, to assess hypotheses and to predict unknown values of the location. Fischer (2006) explains that SDA identifies the characteristics of spatial data, finds the patterns and relationships by testing with hypothesis and models. Goodchild et al. (1992) state by referencing Ripley (1981), and Getis and Boots (1978) that the range of SDA techniques covers “from simple descriptive measures of patterns of events to complex statistical tests”. Goodchild et al. (1992) also add that SDA “does not include statistical techniques that use only the attributes of objects”.

The term “spatial analysis” is mostly used in GIS field, because spatial analysis has a history in geography that goes to the 1950s (Haining 2003). Definitions of spatial analysis made by Haining in Fotheringham and Rogerson (1995) and Haining (2003) are similar with the definition of SDA. However, Fotheringham et al. (2000) state that the meaning of “spatial analysis” has changed in time by the GIS vendors. GIS vendors advertised that their GIS software products have advanced spatial analysis tools; however, these tools perform mostly geometric operations and manipulation on spatial data. The term “spatial analysis” and SDA can be used as the similar meaning and are interchangeable words in some cases.

The difficulty of explaining SDA comes from the close meaning of spatial analysis. There is a particular distinction between spatial analysis and SDA which is clarified by Bailey and Gatrell (1995). SDA tries to explain the spatial processes and the relationships between other spatial phenomena. In order to emphasize this distinction, in this thesis, the term, Spatial Data Analysis (SDA) is adopted instead of using spatial analysis.

Many areas have contributed to the development of spatial analysis and similarly to SDA. When the historical development of spatial analysis is reviewed, the trace of the fields can be seen in many different application areas. Longley and Batty (1996) state that spatial analysis mainly belongs to quantitative geography, the ecology, transportation and urban studies. Also, other areas helped to develop this field. SDA techniques have been developed in various application areas such as geography, biology, epidemiology, statistics, geographic information science, remote sensing, computer science, mathematics, scientific modeling, etc. Many areas have caused to develop different style of analysis. Bailey and Gatrell (1995) explain the meaning of this variety as that some of existing SDA techniques are re-invented.

Wise and Haining (1991 cited by Haining in Fotheringham and Rogerson 1995) classify the spatial analysis in three main categories; SDA, map based analysis, and mathematical modeling. Although this categorization of spatial analysis is different from Fischer’s classification that spatial analysis has two major areas namely; SDA and spatial modeling, the classification of spatial analysis into three groups is more convenient, because the map based analysis tools in a GIS environment should not be included in the SDA field.

2.2. Types of Spatial Data Analysis

According to the Bailey and Gatrell (1995), Spatial Data Analysis (SDA) can be structured under four subjects:

1. *Point Pattern Analysis*
2. *Spatially Continuous Data Analysis*
3. *Areal Data Analysis*

4. *Spatial Interaction Data Analysis*

First subject in SDA is the point pattern analysis. Bivand et al. (2008) defines a point process as “a stochastic process in which the locations of some events are observed in a closed region”. Point pattern analysis investigates the proximity of the events and their locations in relation to each other. This analysis deals with the spatial configuration or arrangement of the events (Bailey and Gatrell 1995). The benefit of the analysis is that if the clustering is found, then which spatial and/or temporal processes cause this significant pattern are started to be researched. For example; quadrat methods, kernel estimation, nearest neighbor distances, the K function, complete spatial randomness and their tests, space-time clustering and clustering around a specific point source are some of SDA techniques in the point pattern analysis.

Second one is the continuous measurement for point locations, which is known as spatially continuous data analysis. This analysis is mainly used to estimate the values at unmeasured locations. Continuous surfaces such as soil, climate, geology, air pressure, temperature, soil acidity, etc. are modeled according to this analysis. The techniques like various types of kriging analysis such as ordinary, simple, global, block, co-kriging, etc., whose result produces a spatially continuous data are mostly accepted as in the content of geostatistics field. For example; spatial moving averages, tessellation methods, kernel estimation, covariogram and variogram, trend surface analysis, generalized least squares, models for variograms, simple kriging, general spatial prediction, ordinary and universal kriging, block kriging, co-kriging, principal components, factor analysis, principal coordinates and multidimensional scaling, procrustes analysis, canonical variates and canonical correlation are example techniques for the analysis of spatially continuous data.

Third subject is areal data. The problem of this area is to understand the spatial arrangement of aggregated values in areal unit, to detect patterns, and to examine relationships among the set of variables. This subject deals with whether the sets of neighboring areas tend to have the same similar values or not. For example; proximity measures with area data, spatial moving averages, median polish, kernel estimation, spatial correlation and the correlogram, generalized least squares, tests for spatial correlation, spatial regression models, probability mapping, empirical bayes estimation and generalized linear models are some of the SDA techniques used to analyze areal data.

The last subject is the spatial interaction data which refer to the data on flows that link a set of locations. Flow is a realized movement of people, goods, services or information between an origin and a destination. The objective of this analysis is to model the pattern of flows in terms of the geographical accessibility, the demand at origins and the attractiveness of destinations (Bailey and Gatrell 1995). For example; gravity models, location-allocation problems and network problems are example ones for the analysis of spatial interaction data.

In addition to these, SDA also deals with the relationship between the attributes of the spatial objects and the spatial arrangement of the object. Bailey and Gatrell (1995) explain that the information for both locations and attributes is required as a minimum in order to perform SDA techniques. When only the attributes are statistically analyzed, this kind of analysis cannot be accepted as a part of SDA. If the attribute data are detached from spatially referenced objects, the attribute will lose its value and the meaning of spatial context will disappear. If the aim is to compare the arrangements of different types of entities, the variables describing the spatial objects as the attribute value can be considered as the part of SDA. In contrast, when the aim of the research area is only to study the spatial arrangement or the pattern of the spatial objects, there is no need to collect any attribute information about the location.

2.3. Coupling of SDA Software with GIS

There is a coupling problem of SDA with GIS, which means that SDA techniques should fully be integrated with GIS, because SDA requires very tedious computation processes and GIS offers a potentially valuable platform for SDA techniques. Both SDA and GIS will benefit from this integration. SDA should benefit from the data input, storage, retrieval, data manipulation and display capabilities of GIS. This integration also serves for proper tools of disseminating the SDA techniques. GIS can also benefit from SDA techniques in which, the integration of these techniques can make GIS to explore, analyze and predict the behavior of the spatial processes. This will lead to increase analytical capabilities of GIS. SDA techniques can accelerate the progress of analyzing spatial processes of GIS.

There is a widespread agreement on the benefits of coupling of SDA with GIS (Wise and Haining 1991 cited by Haining in Fotheringham and Rogerson 1995). The clear contribution of GIS to form SDA can be considered as the manipulation, integration and exploration of spatial data (Fotheringham et al. 2000). Without a system and associated SDA software, the initial costs for analyzing spatial data can be unaffordable (Haining in Fotheringham and Rogerson 1995).

Academic community sees that current GIS functionality is very limited in terms of providing SDA techniques. The integration of SDA in GIS will also extend the utility of GIS. Goodchild et al. (1992), and Fotheringham and Rogerson (1995) have complained about the lack of analytical power of GIS. Goodchild (1987) observes that GIS places more emphasis on efficient data input and retrieval than on sophisticated analysis (cited by Gatrell and Rowlingson in Fotheringham and Rogerson 1995). GIS clearly needs stronger analysis and modeling capabilities (Goodchild et al. 1992). The wider research scientific community can use SDA techniques, when they are implemented in the GIS software. In addition to these, the most important contribution of integrating SDA techniques in GIS software is the use of GIS with spatial models (Fotheringham et al. 2000).

SDA researchers have tried to find an appropriate way in order to succeed the complete integration of SDA techniques in GIS software. After all discussions, Goodchild et al. (1992) formulated their integration approaches into four types of coupling strategy as follows:

1. *Loose coupling of proprietary GIS software with statistical software*
2. *Close coupling of GIS and statistical software*
3. *Standalone spatial analysis software*
4. *Complete integration of statistical spatial analysis in GIS*

First type of coupling problem is the loose coupling of proprietary GIS software with statistical software (Goodchild et al. 1992). Since the fourth strategy is unrealistic from vendors point of view, most attention focuses on the question of whether loose coupling or close coupling is desired (Fotheringham and Rogerson 1995). One of the approaches to the problem may be referred to as loose coupling between the GIS software and statistical package (Bailey and Gatrell 1995). In the loose coupling, some kind of interface or reasonably seamless link is sought between statistical package and GIS software (Fotheringham and Rogerson 1995). While this offers some flexibility, it is inefficient (Bailey and Gatrell 1995). One way of doing conversion is to use ASCII files. The other way is to convert to common spatial binary format. The data conversion is done by using importing and exporting tools. Spatial data are exported to common format by GIS software. The format is read by statistical package. It can be needed to make corrections in the structure of the data in the package. After completing the analysis in the package, if there is a spatial output, the result is exported to GIS data format in order to overlay as a map with other spatial layers in the GIS software. Fotheringham et al. (2000) state that there is no reason why this conversion should be done at the same time.

Second type of coupling problem is the close coupling of GIS and statistical software. In close coupling, SDA theorists might seek to embed statistical analysis functions into GIS (Fotheringham and Rogerson 1995). Close coupling is a better strategy then loose coupling, because it involves calling spatial analysis routine from within the GIS (Bailey and Gatrell 1995). SDA techniques are implemented within the GIS by scripts or macro language of GIS software. This kind of development work can be called as the creation of an extension, add on, plug-in, or package. SDA researchers demonstrated some integration examples in order to convince the GIS vendors that this integration could be easily made. The problem is that this integration is done by the SDA community not by the GIS vendor. These attempts have not convinced the GIS vendors. They did not include the algorithms of SDA techniques inside the GIS software.

The close coupling strategy did not work, when the architecture of the GIS software has changed with the next revision. For example, DynESDA is a SDA extension for ESRI ArcView GIS series 3 and will never run in ESRI ArcGIS, because the architectures of ArcView GIS series 3 and ArcGIS are totally different. From the GIS vendor's point of view, the commercial GIS software has become so much complicated that none of the vendor wants to increase this complexity by adding more tools that will be rarely used by the users. This trend has started to change with the development of GIS and with the changes of the needs of the users. GIS users have started to request to do more advanced spatial researches.

Third type of coupling strategy is the development of standalone SDA software which is mainly developed by SDA theorists. Software maintenance is required in order to keep the software to work properly. As much as 80 percent of software costs are related with the on-going maintenance of the software throughout its lifetime (Deitel et al. 2002). The developing and maintaining the standalone SDA software is too much costly and requires spending a lot of time to have full featured program. The needs of users, the structure of GIS data formats, external dependencies related with utilities, libraries, programming languages and operating system can change in time which leads to break down the architecture of the SDA software. New and revised versions of GIS data formats, standards and protocols can be emerged in the GIS environment. This kind of possible changes and improvement requirements makes the software maintenance very tedious and costly job. It is very difficult to renew the software especially when radical changes have occurred in the operating system or in one of the external dependencies of used library.

Performance of the software while working especially with spatial data has become one of the most important factors. The performance of the software depends on the architecture of software. In order to upgrade the performance of the software, new technologies and architectural innovations should be included in the SDA software. In many cases, SDA theorists' efforts do not overcome the difficulty of doing continuous development and maintenance work. This task is very hard to achieve solely by the SDA theorists. Without applying these improvements, the architecture of SDA software will become out-dated and have performance constraints. Much SDA software has stacked with these difficulties. In the long term period, the standalone SDA software products like Info-Map, SpaceStat and GeoDa have become out-dated due to not being upgraded according to the changes in the operating system.

Developed standalone SDA software is mostly designed to work for Microsoft Windows Operating System (OS) which is a closed source environment. Microsoft has the power of doing any changes in his operating system. When this monopolistic power is applied on behalf of the company interest like not including one of dependencies of GIS libraries in operating system, the SDA software may become out-dated. The architecture of the new revision of operating system has sometimes changed so dramatically that the efforts to adapt these changes could not be done solely by several SDA developers. In addition to this, applying such modifications from the vendor is unrealistic, because there is weak demand from the market and it is not much profitable for GIS vendor. This is one of the weaknesses of the closed source environment in terms of sustainability of standalone SDA software without the support of market for SDA field.

When the SDA software is designed to be developed with the closed source software development methodology, third party developers could not contribute for the development of SDA software. Developing any algorithm in the main source code of the proprietary software by external developer is almost impossible in the closed source environment, since there is no freedom to use or even to see the piece of source code of proprietary software. This is another weakness of the closed source environment.

Another problem of such an integration approach in the closed source environment is that the user has to rely on the manner how the software vendors have implemented the SDA techniques in their software. The user does not have a right to examine how these SDA techniques are running from the source code of the tool. The algorithm of techniques could not be accessed by the user of the software. In some cases, out of the box approach creates more severe problems. Fotheringham et al. (2000) state that although the same SDA technique is used in different software, the results could be different with the same data.

In addition to these difficulties, when the third type of coupling strategy is followed, the capabilities of GIS such as data entry, editing, display, mapping, reading and writing different GIS data formats, management of coordinate systems, connection with database systems, performing geometrical operations, etc., should be reinvented from the scratch which leads to increase the development cost and required time for the development of the software. While the GIS software has succeeded to become predominant software in the geospatial industry, SDA should benefit from the capabilities of GIS. In the third coupling strategy, it is required to re-invent the whole capabilities of GIS.

Fourth type of coupling problem is the complete integration of SDA techniques in GIS. This integration requires the participation of the GIS vendor which should incorporate SDA techniques within their software products. But, their main motivations are in the utilities and local government rather than in the SDA research community (Fotheringham and Rogerson 1995). It is unrealistic to expect from the

GIS vendors to make this integration, because these areas are more profitable for GIS vendors than the scientific research (Bailey and Gatrell 1995). The main reason is that there is a weak demand from the market for the use of SDA techniques in GIS. The integration request has only come from the scientific research community, but the request is gradually increasing at the last decade.

This integration approach is eliminated from the initial stage of GIS software development process. Most of the SDA techniques were developed when the GIS is at the initial stage. In 1960s, the urgent need of GIS is to perform the geometry related operations such as overlay, point-in-polygon, intersection etc. At that time, the computation power of computer was very limited. For this reason, the GIS vendors could not deal with SDA techniques.

Bailey and Gatrell (1995) add that some researchers have taken statistical package and have developed SDA techniques in that package. But, the package has limited GIS functionality. Analyzing spatial data in statistical package may cause misleading results, because the analysis may not be valid when the variables are spatially correlated with each other. According to the fundamental statistical methods, the following steps are performed; in the first stage, a scatter plot of searched variables against each other is drawn, and then, if there is a reasonable degree of association, a correlation coefficient is calculated and statistical significant test is applied. Finally, the relationship between them is explained by fitting regression equation. This kind of fundamental statistical method should be applied to only the attributes of spatial data, because spatial data often have special properties and need to be analyzed in different ways from tabular data. The main problem for the integration of SDA in the statistical systems is that the spatial data are mostly analyzed from the statistical perspective in this environment. The space can make a difference and should be interpreted in a different environment.

Some of the SDA tools have historical roots in 1960s and were continuously developed since then. At past, the way of integrating SDA with statistical systems was tried, because at that time GIS was in the early stages and struggling with the geometry related problems. This approach was seen as a kind of compulsory way due to the limitations of the computers for the geography related systems in 1960s. Several modules or extensions for this integration were developed in statistical packages such as SPSS, Minitab, S-plus etc. That’s why; statistical software vendor states that their software also does the spatial data analysis, which is true in some sense. However this kind of integration is the same as “loose coupling”, because the geography related data formats should be transformed from GIS application to statistical package. It is impossible for statistical packages to have all kinds of GIS tools. For this reason, in addition to the spatial auto-correlation problem, the integration of SDA with statistical systems is not the proper way. That’s why; this approach is eliminated from the beginning in the classification of the coupling problem.

The SDA techniques require the knowledge of statistics and are explained with sophisticated mathematical equations, which makes difficult to be understood by the GIS users. Even many academic geographers, who have limited quantitative and scientific backgrounds, have perceived SDA techniques to be relatively difficult. The applications of SDA techniques require very intensive mathematical calculations. The programming of such mathematics into real application is a work that a few researchers can handle. For this reason, some geographers lost interest in quantitative work when it became too mathematically demanding (Fotheringham et al. 2000). This is one of the constraints that avoided the dissemination of SDA.

There are several reasons why coupling of SDA with GIS cannot be accomplished. The market has played a major role in the development of GIS software, which is mostly interested in resource, infrastructure and facility management, and land information rather than scientific research. However, mostly academic and scientific communities were interested with SDA field since 1960s. Goodchild et al. (1992) explain that “SDA remains an obscure field”, which makes very difficult to organize the SDA field. Many such techniques were originally developed outside of the field of statistics; for example in geography, geostatistics, econometrics, epidemiology, or urban and regional planning and the wide range of relevant literature reflects this (Fotheringham and Rogerson 1995). This range of variety complicates to follow the SDA techniques in which some of them can be found in one application area and others are in another area. Development of SDA techniques has started before GIS has evolved into a mature stage (Goodchild et al. 1992). GIS was not mature enough in computer system to integrate these techniques in 1960s and 1970s. The lack of enough computational power for processing spatial data is another problem for GIS field at that time. SDA techniques require additional programming effort in the computation environment. In 1970, statistical packages such as SAS and SPSS have integrated SDA routines in their applications. However, integrated environment does not mainly support the coordinate systems and the spatial objects. This environment has limited functionality such as mapping and displaying of spatial data. In addition, statistical environment is non-spatial and there is spatial autocorrelation problem between the events located in the space. In the statistical environment, it is tried to avoid this problem while analyzing the spatial data, but, Goodchild et al. (1992) explain that this problem should be accepted as “an inescapable property of spatial data”.

As a result, in order to succeed the complete integration of SDA techniques in GIS, a chance to modify the source code of the SDA techniques or to add new SDA tools to GIS software by different SDA developers should be possible. Many SDA techniques are developed and used in different application areas. These kinds of range and varying interests in SDA have meant that many analysis tools have been developed independently and with their own distinctive styles of analysis. This kind of variety causes different terminology and usage. Some tools are implemented for specific model and data driven systems. This means that the implementation of SDA tools are scattered throughout different fields. For this reason, in order to integrate SDA techniques in GIS, more collaborative software development methodology is required. With FLOSS development methodology, SDA researchers and developers coming from different application areas can work together to develop the software. Other factors like being cross platform, user friendly, easy to use, free in terms of cost to use software and interoperable with the Open Source Geospatial Consortium standards should also be solved in the same development environment.

2.4. Overview of Existing SDA Software

There are a lot of niche SDA software products that have limited in number of SDA tools and GIS functionality. These software products specific to one application area like crime, health, etc. are not considered in this thesis. The SDA extensions, standalone and generic SDA software are reviewed as shown in Table 2.1. These software examples are studied according to four subjects; standalone SDA software, close coupled SDA extension, proprietary desktop GIS software and FLOSS product.

Table 2. 1. SDA Software Examples and Their Executable Status

Type of Software	Software	Advantages	Disadvantages	Executable status
Standalone SDA Software	**Info-Map**	software package distributed with a book for interactive spatial data analysis	closed source package and DOS-based product	quite out-dated; antiquated architecture and performance constraints
	SpaceStat	Provides tools for creation of spatial weights matrices, exploratory SDA, spatial econometric analyses.	closed source package and Windows-based product	quite out-dated; antiquated architecture and performance constraints
	GeoDa	free and user friendly software, supporting interactive exploratory SDA	closed source package and Windows-based product	running on only Windows XP operating systems, quite out-dated
	CrimeStat	spatial statistics program used in crime mapping, links with most of desktop GIS software like ArcGIS and MapInfo	closed source package and Windows-based product, no display capability	running on recent operating systems
Close Coupled SDA Extension	**SpaceStat Extension**	Provides tools for exploratory SDA, an extension program for ArcView GIS series 3	closed source package and close coupling	quite out-dated; antiquated architecture and performance constraints
	DynESDA Extension	Dynamic exploratory SDA, an extension program for ArcView GIS series 3, brushing and linking maps	closed source package and close coupling	quite out-dated; antiquated architecture and performance constraints
Proprietary Desktop GIS Software	**ArcGIS**	Advanced Desktop GIS program, full rich of GIS tools, worldwide used	closed source package and very expensive product	running on recent operating systems
FLOSS Product	**R**	An FLOS statistical package, have many tools in variety of fields including SDA, cross platform support	Not user-friendly environment, requires programming	running on recent operating systems
	GRASS GIS	FLOS GIS software, cross platform support	A few SDA tools, Unix style usage, different data format	running on recent operating systems

2.4.1. Standalone SDA Software

First reviewed standalone and generic SDA software example is Info-Map, which is developed in 1995 by Bailey and Gatrell to analyze spatial data. It is not a fully functioning GIS package in the sense of providing a wide variety of spatial data manipulation options. However, it possesses a wide variety of SDA tools, controlled through a simple command line interface. Info-Map has some simple mapping facilities for the exploration of data and the display of the results of the analysis (Fotheringham et al. 2000). Info-Map is remarkable example which runs only on the Microsoft DOS operating system. However, the software can perform the following SDA techniques; kernel estimation, K functions, variogram estimation, kriging, spatial autocorrelation analysis, spatial regression, principal components analysis, multivariate classification and non-spatial regression. It is surprising that all these tools are available in 1995 and working only in the Microsoft DOS operating system. However, today, Microsoft DOS operating system does not exist anymore and nobody can use this operating system properly. So, the most predominant factor for the dissemination of SDA is not only the creation of the software but also the provision of the sustainable software development methodology, especially in which there is a weak support from the market for the field that the development will be carried out. Info-Map software cannot be used now, because the originators did not upgrade the architecture of the software in parallel with the changes in the operating system. The software should continuously be developed, maintained and upgraded by the originator of the programmer. Without the support of market, the durability and availability of SDA software cannot be achievable in the closed source environment. That's why; the SDA software should not be performed to develop in the closed source environment.

Second reviewed standalone SDA software is SpaceStat, which is developed by Anselin with the support of a number of grants from the U.S. National Science Foundation in 1983, 1986, 1988, 1990, 1994 (Figure 2.1). Anselin (1992) explains that SpaceStat is first released in 1992 by US National Center for Geographic Information and Analysis. The updated version of SpaceStat is 1.80 in October 1995 and 1.90 in 2000. SpaceStat is purely a 32 bit DOS product, which runs in Windows 95, 98 and NT. SpaceStat version 1.91 has a command line interface like Microsoft DOS operating system in 2002. The SpaceStat Extension requires ESRI ArcView series 3, but SpaceStat itself does not, which is completely self-contained.

SpaceStat is developed according to the traditional software development method and designed to work for only closed source environment, which means that the development of software is not opened to everyone, but can be done only by the originator. When the source code of the software is not opened to external developers, it is becoming nearly impossible to continue the development process in which stage is left. However, developing the software within the open and collaborative environment may guarantee to continue to develop the software. The architecture of the software is structured at the beginning and may evolve in time during development process. With the help of the collaborative development environment, external developers can follow the progress of development work from the changes in the source code. By this way, they can learn how to develop the software like the originator. This collaborative environment can be provided by setting up required tools in the software development project if the project opens the source code of the software as well as the changes in the source code to be accessed through the Internet to the external developers.

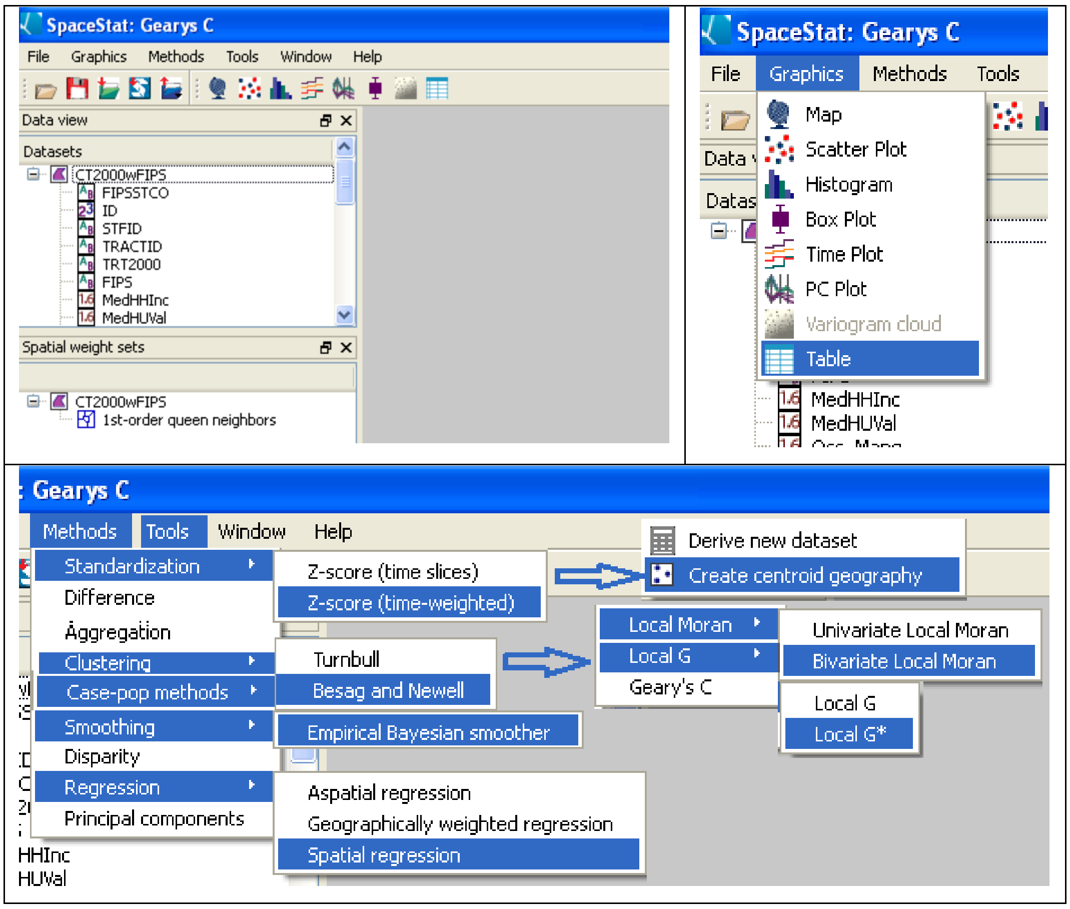


Figure 2. 1. SpaceStat Interface and Analysis Tools

Another standalone SDA software example is GeoDa (Figure 2.2). GeoDa has a user-friendly, graphical and menu driven interface. It is aimed to be developed for non-GIS users. This software is being developed by Anselin since 1999. GeoDa is running on Windows operating system version XP. The disadvantage of this software is that GeoDa is designed to run only closed source environment, which means that it will be in danger of becoming out-dated like Info-Map, SpaceStat, or DynESDA. GeoDa has already struggled with such a problem in 2007.

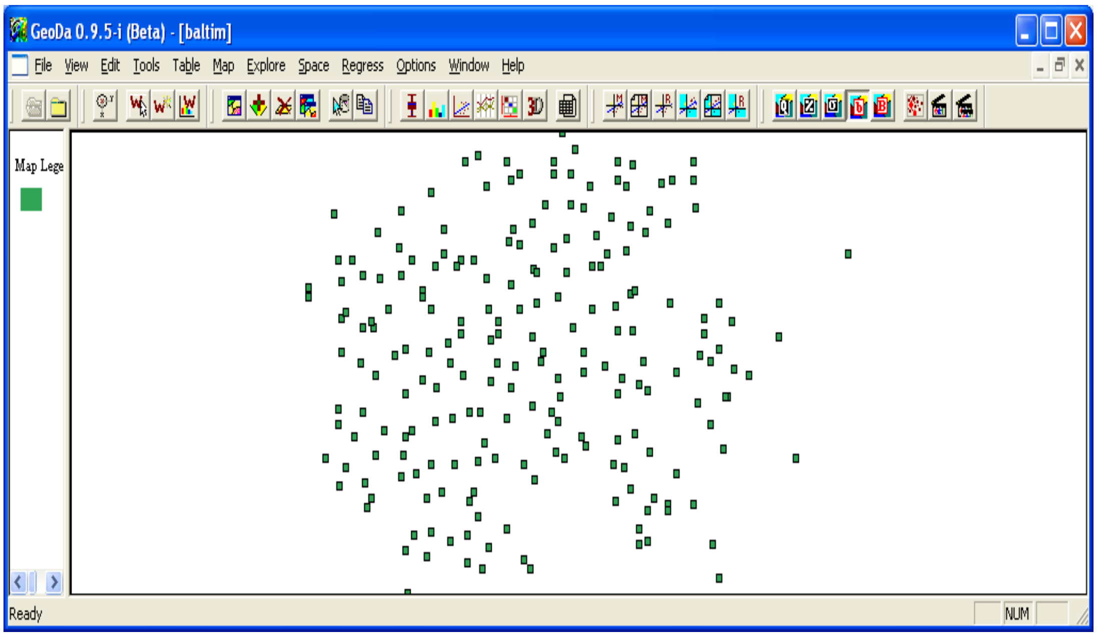

GeoDa 0.9.5-i (Beta) - [baltim]
File View Edit Tools Table Map Explore Space Regress Options Window Help
Map Lege
Ready
NUM

Figure 2. 2. Snapshot of GeoDa Version 0.9.5-i

Anselin et al. (2006) explain that “GeoDa is a reinvention of the SpaceStat”. SpaceStat package has become out-dated with its antiquated architecture. GeoDa is the successor to and a replacement for the DynESDA extension coupled with ESRI ArcView GIS series 3. GeoDa is freeware standalone software. It is built on ESRI's MapObjects technology and uses most common ESRI Shapefile format as the standard for storing spatial information (GeoDa 2008a). The cost of re-writing of GeoDa software from scratch is financed by a Center for Spatially Integrated Social Science, which is founded with a grant provided by US National Science Foundation. The awarded amount was $4,335,573 (NSF 2007). The project was initialized in 1999, finalized in September, 2004, and extended one year at no cost. GeoDa project is a five year research infrastructure project and promotes an integrated spatial approach to social science researches (CSISS 2002). The center was established in 1999. The aim of the center is to establish an infrastructure for social sciences with a spatial analytical perspective (Goodchild 2000). One of objective of the project is to develop a powerful and easy to use software doing SDA (CSISS 2002).

The first official release of GeoDa version 0.9 was in February, 2003. The bug free or stable version of the GeoDa was announced in September, 2004. The development methodology of this software is for the closed source environment. GeoDa version 0.9.5-i can work only on Microsoft Windows operating system version XP. During the development period of GeoDa, the version of operating system is Windows XP. In a closed source environment like Windows operating system, sustainability of such a development process is not strong enough to sustain the software without continuous support of any vendor or community. As average in every 2.18 years, Microsoft develops a new version of operating system which means that radical change may occur in the structure of operating system's libraries within this period (Table 2.2). In this case, when GeoDa has reached the stable version at such a time that the support of fund was completed. If one of geography related library is not added in the next new version of the operating system, the program will not function properly, because the cost of adapting such a modification is insuperable when especially the support of fund is cut and there is no support from commercial vendor or community. The cost of keeping the software functional can be unmanageable without the support of any vendor as average in every 2.18 years in Windows operating system. This phenomenon, becoming out-dated, is very common for SDA software like Info-Map, SpaceStat and DynESDA that is designed to work only in Microsoft environment. Much SDA software working only in Microsoft closed source environment has become quite out of dated. Under a closed source software development methodology, the responsibility of either upgrading or updating the architecture of the software belongs to only the originator of the software, which is one of the constraints of the closed source development system.

Table 2. 2. Release Date of Microsoft Windows Operating System (Microsoft 2011)

Windows Versions	**Based on**	**Release Date**	**Release Periods**	
			In Days	**In Years**
Windows 1.0	DOS	20.11.1985	-	-
Windows 2.0	DOS	09.12.1987	749	2,05
Windows 3.0 (DOS 3.1)	DOS	22.05.1990	895	2,45
Windows NT 3.1	NT Kernel	27.07.1993	1162	3,18
Windows 95 (DOS 7.0, 7.1)	DOS	24.08.1995	758	2,08
Windows NT 4.0	NT Kernel	24.08.1996	366	1,00
Windows 98 (DOS 7.1)	DOS	25.06.1998	670	1,84
Windows 2000 (NT 5.0)	NT Kernel	17.02.2000	602	1,65
Windows Me (DOS 8.0)	DOS	19.06.2000	123	0,34
Windows XP (NT 5.1 & 5.2)	NT Kernel	25.10.2001	493	1,35
Windows Vista (NT 6.0)	NT Kernel	08.11.2006	1840	5,04
Windows 7 (NT 6.1)	NT Kernel	22.10.2009	1079	2,96
		AVERAGE	**794,27**	**2,18**

The same situation becoming out-dated has happened also for GeoDa software. The first complaining e-mail about GeoDa that could not work on Windows operating system version Vista was in March, 2007 (Bobholz 2007). The experts have tried to find the reason why GeoDa did not work on Windows Vista during two months. The problem is related with the MapObjects-Java libraries which are not included deliberately by Microsoft in the Windows Vista. Not only GeoDa but also ESRI's products rely on the MapObjects-Java libraries. ESRI explains the problem with an announcement in May, 2007 that there are known issues with Windows Vista that ESRI ArcGIS version 9.2 does not support Vista. Then, next announcement was done in December, 2007, which explains that Windows Vista could not be supported until the next release of ArcGIS version 9.3 (ArcGIS Resource Center 2007). The problem is so much severe that even the power of ESRI is not enough to solve the problem of not running all ArcGIS products version 9.2 in Windows Vista due to not including MapObjects-Java libraries in the next revision of operating system after Windows XP. ESRI can only fix the problem with the release of next new version of their product. ESRI ArcGIS products have supported Windows Vista with the release of version 9.3 in June 2008. Not including such an important GIS related library can only be done by the Microsoft, who has the power of monopoly in such an environment.

In order to fix this problem, Center for Spatially Integrated Social Science has started to implement FLOS wxWidgets cross-platform GUI library to GeoDa instead of using proprietary library, MapObjects. This maintenance work is named as creating a new software package OpenGeoDa. By means of using wxWidgets library, GeoDa has become to support the cross-platform. So, OpenGeoDa can run Windows (XP, Vista and 7), Linux (with both 32 and 64 bit) and Mac OS X (Intel and PPC). Current released version of OpenGeoDa in August, 2010 has reached to alpha release 0.9.8.14. The meaning of alpha version is that the software may not be free of errors. After alpha release, there is beta release, and after then, the software could be reached to stable version. This alpha release contains the functionality of GeoDa 0.9.5-i (GeoDa 2010b). Fixing the breakdown of GeoDa on Windows Vista by upgrading the architecture of GeoDa to FLOSS environment is required to spend more than three and half year. This maintenance work could not be completed in August 2010, since the stable version of OpenGeoDa has not been released yet. GeoDa can work in Windows operating system only in two and half year between the date of stable version of GeoDa in September 2004 and the date of first complaining e-mail in March 2007.

Anselin and McCann (2009) advertise that OpenGeoDa is the FLOS successor to GeoDa (Figure 2.3). This is true in some sense, but, the main distinct component is missing; having a freedom to access the source code of software. Without providing this requirement, OpenGeoDa software could not be accepted as FLOSS product.

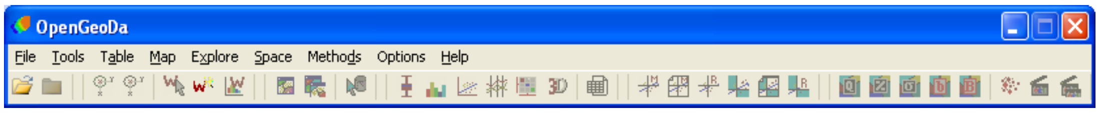


Figure 2. 3. Snapshot of OpenGeoDa

According to the research done in August 2010 through Internet, the following traces are found about OpenGeoDa. It is hoped that these traces are the sign that OpenGeoDa will be FLOSS product of SDA field in the near future. OpenGeoDa is registered as FLOSS project in the SourceForge.net at the end of 2005, but no activity can be seen since then (OpenGeoDa 2010a). There is an e-mail list named as Openspace List, in which discusses all aspects of developing methods for spatial data analysis and their implementation in the form of FLOS software tools with the users of GeoDa (OpenGeoDa 2010b). The last trace is to use the project hosting infrastructure of Google Code in order to inform the progress and problems during the development process of OpenGeoDa since September 2009 (OpenGeoDa 2010c). Although subversion check-in and check-out operation is defined in the OpenGeoDa of Google Code, the source code is removed from this repository. Source code files could not be obtained when read-only check-out operation is completed successfully with the subversion application. The repository of OpenGeoDa could not be accessed through the Internet in August, 2010 from FLOS collaborative development environments; SourceForge.net and code.google.com. In every valid FLOSS project, there is information which explains how to access to the source code of the software through the Internet. In addition to these, OpenGeoDa is distributed as only one binary file in the official home site of GeoDa (GeoDa 2010b). Distributing with binary file hides the source code of the software from the users. One binary file is advantage for users, since OpenGeoDa does not require any installation and simply works as one executable file, but, this kind of binary distribution prevents the software to be accepted as FLOSS product. OpenGeoDa uses only FLOS GUI library in order to be cross platform and distributed at no cost like any freeware application.

Development work carried out for GeoDa is as the same as Goodchild's motivation to develop user friendly software in 1992.The distribution of GeoDa across the globe in the map is very fascinating (Figure 2.4). The number of GeoDa users has crossed 19,000 (19,058) in between the first official release in February 2003 and March 2007. The figure continues to grow exponentially and has reached to nearly 52,000 (51,988) in August 2010. More than 705 new users downloaded the software every month from May 2006 to March 2007, 360 monthly downloads between December 2004 and July 2005, 220 between September and December 2004, and 170 between May and September 2004. The majority of GeoDa users are affiliated with a university (84%), compared to 11% who work for government agencies (GeoDa 2008b). Although the development of GeoDa has created valuable contributions to the SDA community and made many of the SDA researchers' works easier, the software has become quite out-dated in a short time due to applying wrong software development methodology in 2000s as well as happened for Info-Map, DynESDA and SpaceStat in 1990s.

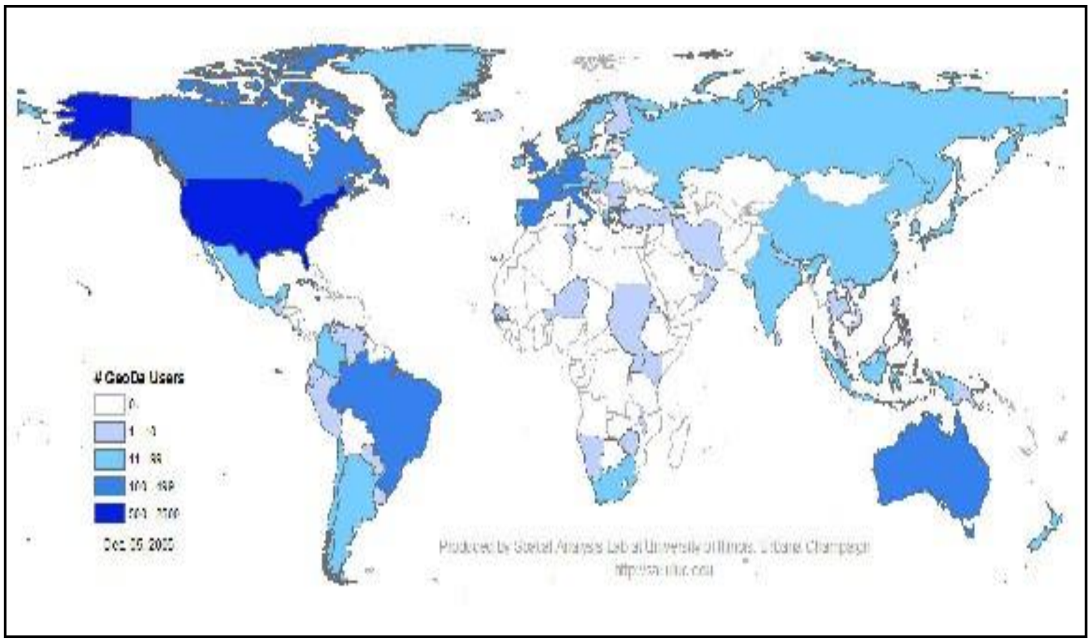
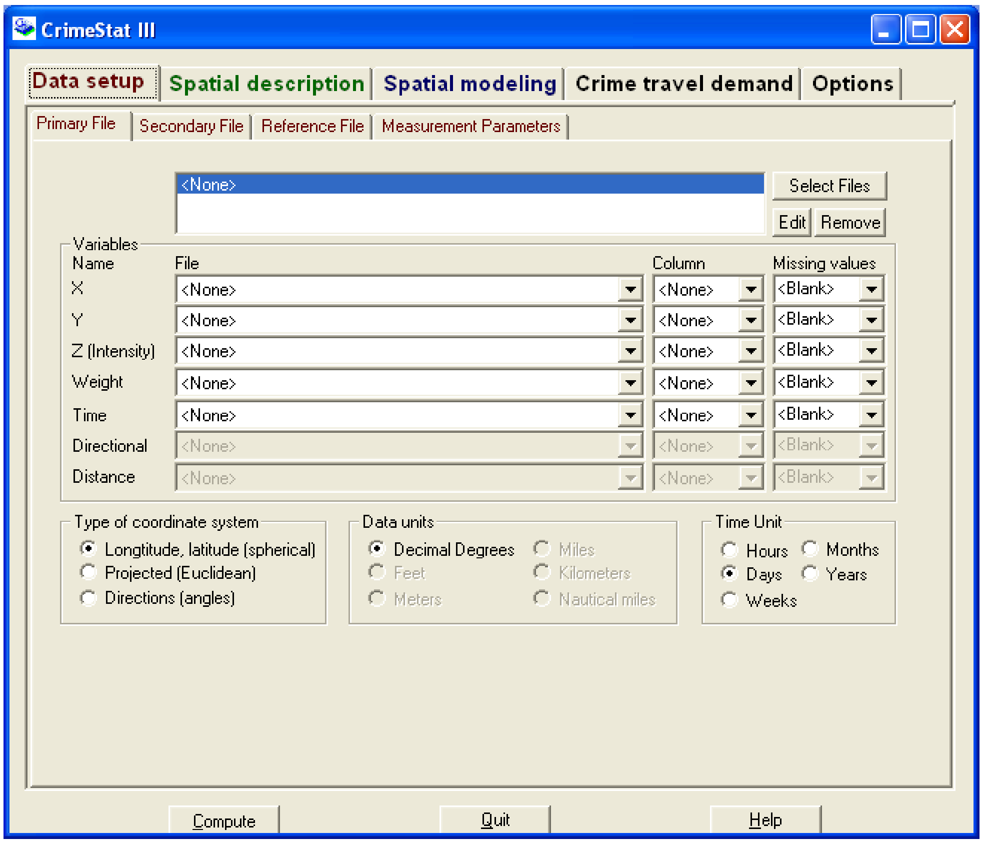

# GeoDa Users
0
1 - 10
11 - 99
100 - 499
500 - 2000
Dec 25 2005
Produced by Spatial Analysis Lab at University of Illinois, Urbana-Champaign

Figure 2. 4. GeoDa Users across the Globe between February 2003 and March 2007 (GeoDa 2008b)

Last reviewed standalone SDA software is CrimeStat (Figure 2.5). It is mainly used for the analysis of crime mapping, but, statistical tools of CrimeStat can also be used in the point pattern analysis. Therefore, this software is accepted as generic standalone SDA software. First release of CrimeStat was done in November 1999 and the last one was in March 2005 (Levine 2006). Levine (2005) explains that the development of CrimeStat was supported three times with grants provided by US National Institute of Justice. The development of latest version 3 is made possible through the support of several public organizations and programs (CrimeStat 2011a). Levine (2006) informs that the purpose of whole development is to provide SDA techniques to assist “law enforcement agencies and criminal justice researchers in the crime mapping works”. Levine (2005) adds that many police departments, criminal justice and other researchers use this software in their works.



<None>

Variables

Name	File	Column	Missing values
X	<None>	<None>	<Blank>
Y	<None>	<None>	<Blank>
Z (Intensity)	<None>	<None>	<Blank>
Weight	<None>	<None>	<Blank>
Time	<None>	<None>	<Blank>
Directional	<None>	<None>	<Blank>
Distance	<None>	<None>	<Blank>

Type of coordinate system: [✓ Longtitude, latitude (spherical)] Projected (Euclidean) Directions (angles)

Data units: [✓ Decimal Degrees] Miles Feet Kilometers Meters Nautical miles

Time Unit: Hours Months [✓ Days] Years Weeks

Figure 2. 5. Snapshot of CrimeStat

Point locations are accepted as input in the software and statistical analysis results are shown spatially as output in other desktop GIS software products (Levine 2006). The software can read “DBF”, “SHP”, ASCII or ODBC-compliant formats (Levine 2005). Output is exported to related GIS data formats that have interfaces with most desktop GIS software products such as ArcView, ArcGIS, MapInfo, Atlas GIS and Surfer for Windows (CrimeStat 2011b).

CrimeStat has five basic groupings with seventeen program tabs and one option tab. The five basic groupings are data setup, spatial description, spatial modeling, crime travel demand modeling and options. Data setup part includes reading the data as input from primary, secondary and reference file. Spatial description part contains the spatial distribution, distance analysis and hot spot analysis. Spatial modeling part is interpolation, space-time analysis and journey to crime analysis. And lastly, crime travel demand modeling part contains trip generation, trip distribution, mode split and network assignment.

Disadvantage of CrimeStat is that it does not have any display capability, but the result of analysis can be exported to common GIS data formats. Another disadvantage is that the source code of all analysis routines is closed, so, the techniques could not be studied from the source code. Also, it is not cross platform and works only in Microsoft Windows operating system. The development of CrimeStat is continuing due to support of US public organization; National Institute of Justice.

2.4.2. Close Coupled SDA Extension

First reviewed close coupled SDA extension example is the SpaceStat extension, which is developed to couple between the SpaceStat standalone SDA software and ESRI ArcView GIS series 3. The extension exchanges spatial data and visualizes the analysis results between SpaceStat and ArcView GIS series 3. Integrated techniques were mainly composed of exploratory SDA tools (Anselin 2000). Spatial data is transferred from ArcView GIS series 3 to SpaceStat for analysis, and the result is sent from SpaceStat to ArcView GIS series 3 for visualization (Anselin and Bao 1997). The SpaceStat Extension adds two menu items to graphical interface of ArcView GIS series 3; “Data” menu is for data transfer (Upper Side of Figure 2.6) and “SpaceStat” menu is for the visualization of the results of the integrated several exploratory SDA tools such as box map, spatial lag bar chart, LISA map, Moran significance map, residual map and predicted values map (Below Side of Figure 2.6) (Terraseer 2010). Script language of ArcView GIS series 3 is Avenue. Both Avenue and C programming languages are used for the development of this extension. C programming language is required to use for specialized functions and organized in a dynamic link library (Anselin 2000).

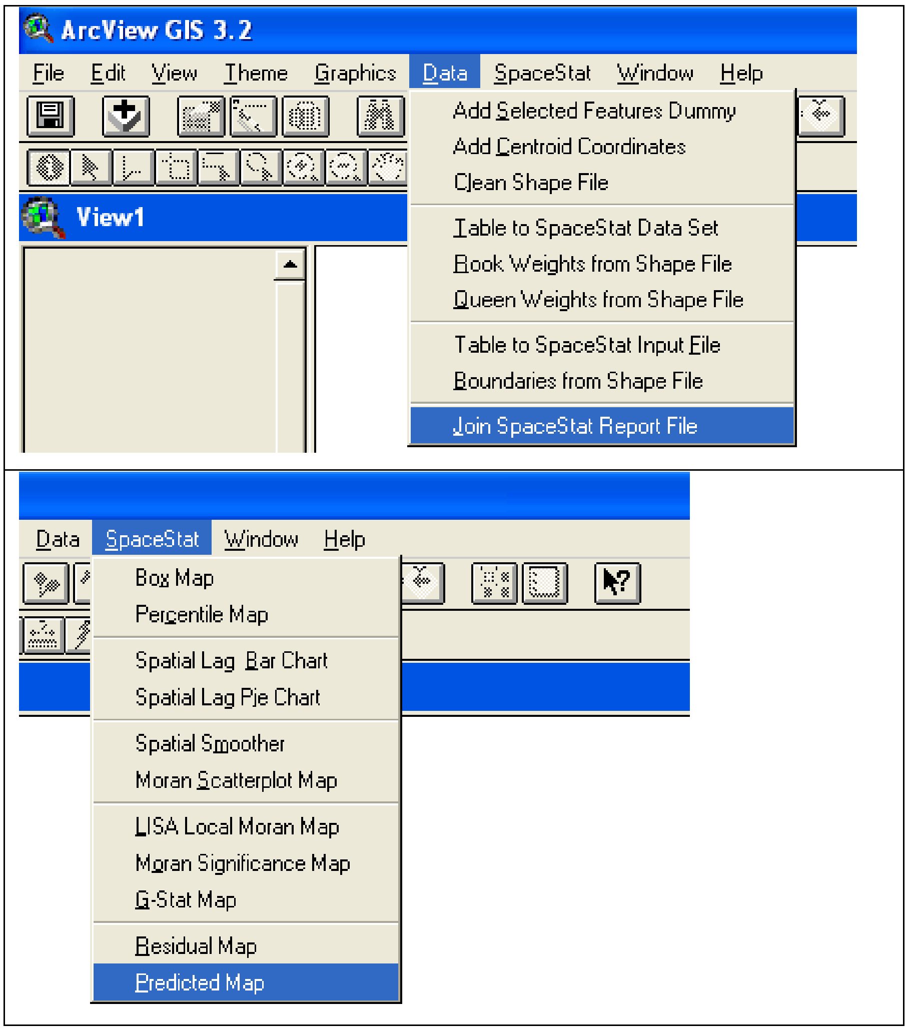

ArcView GIS 3.2
File Edit View Theme Graphics Data SpaceStat Window Help
View1
Add Selected Features Dummy
Add Centroid Coordinates
Clean Shape File
Table to SpaceStat Data Set
Rook Weights from Shape File
Queen Weights from Shape File
Table to SpaceStat Input File
Boundaries from Shape File
Join SpaceStat Report File
Data SpaceStat Window Help
Box Map
Percentile Map
Spatial Lag Bar Chart
Spatial Lag Pie Chart
Spatial Smoother
Moran Scatterplot Map
LISA Local Moran Map
Moran Significance Map
G-Stat Map
Residual Map
Predicted Map

Figure 2. 6. Data (Upper Side) and SpaceStat (Below Side) Menu Items of SpaceStat Extension in ArcView GIS Series 3

Another reviewed SDA extension is DynESDA extension which is developed by Anselin and Smirnov to work in ESRI ArcView GIS series 3. The DynESDA extension for ArcView GIS series 3 was developed according to the idea of dynamic graphics used in exploratory data analysis (Anselin 2000). The extension can produce histogram, box plot, scatter plot, Moran scatter plot and local Moran plot with dynamic linking and brushing between the charts, tables and maps (Figure 2.7) (Matthews 2002). Anselin (2000) defines dynamic linking and brushing as any highlight in the chart or table is also simultaneously highlighted in the map or vice versa. Development purpose of this extension is to realize this dynamic linking and brushing with exploratory SDA tools also in the GIS environment.

ESRI has started to develop new desktop GIS products named as ArcGIS in 1999. The development of ArcView GIS series 3 was stopped in 2002. The software architecture of ArcView GIS series 3 and ArcGIS is completely different. For this reason, SpaceStat and DynESDA extensions could not be continued for further development. These two extensions have become out-dated in time due to structural change in the part of coupled GIS software.

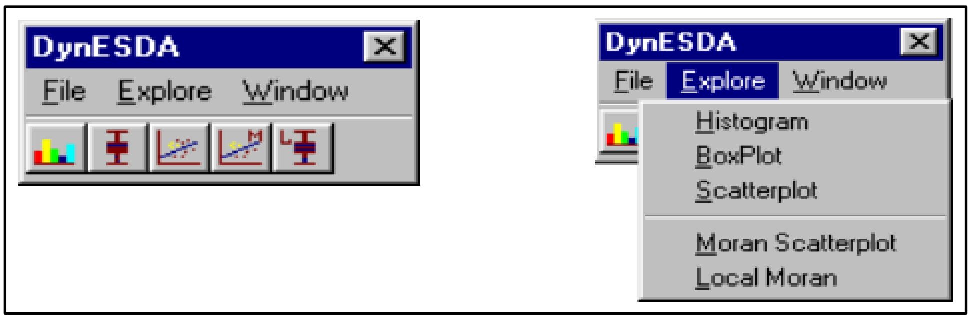


Figure 2. 7. DynESDA Extension in ArcView GIS Series 3

2.4.3. Proprietary Desktop GIS Software: ArcGIS

Most famous proprietary desktop GIS software is ArcGIS which has started to be developed by ESRI since 1999 (Figure 2.8). In late 1999, ESRI released ArcGIS version 8.0 which has a new GUI and composed of several software applications. This package combines ArcView GIS series 3 and ArcInfo workstation version 7.2. ESRI is continuing to develop ArcGIS series as both desktop and server application. ESRI ArcGIS Desktop is one of the most widely used proprietary GIS software products in the world.

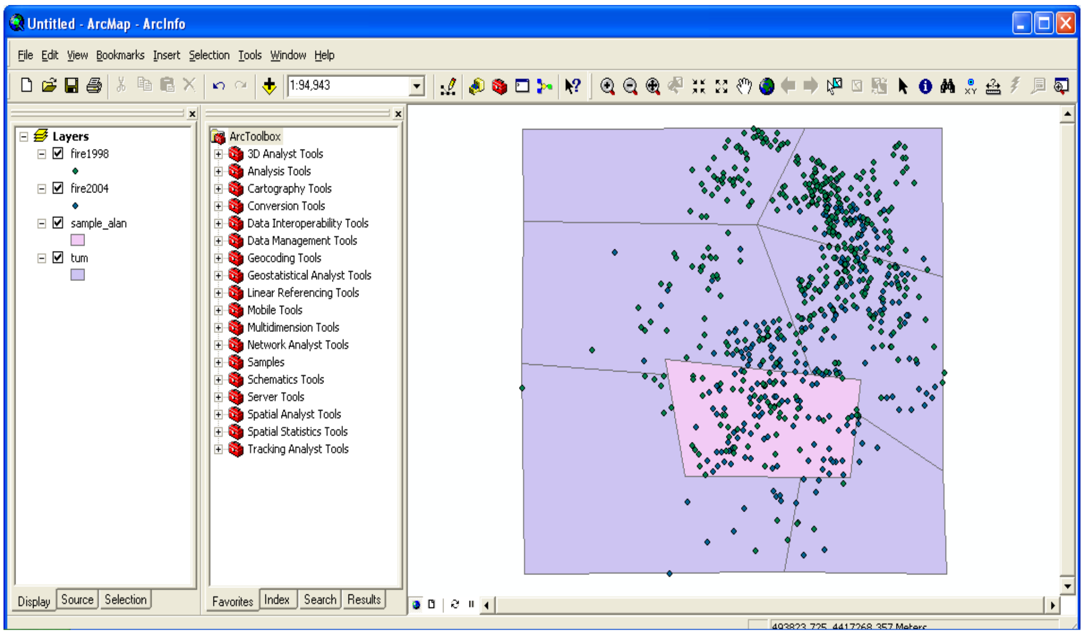


Figure 2. 8. Snapshot of ArcInfo Licensed ArcGIS Desktop Version 9.3

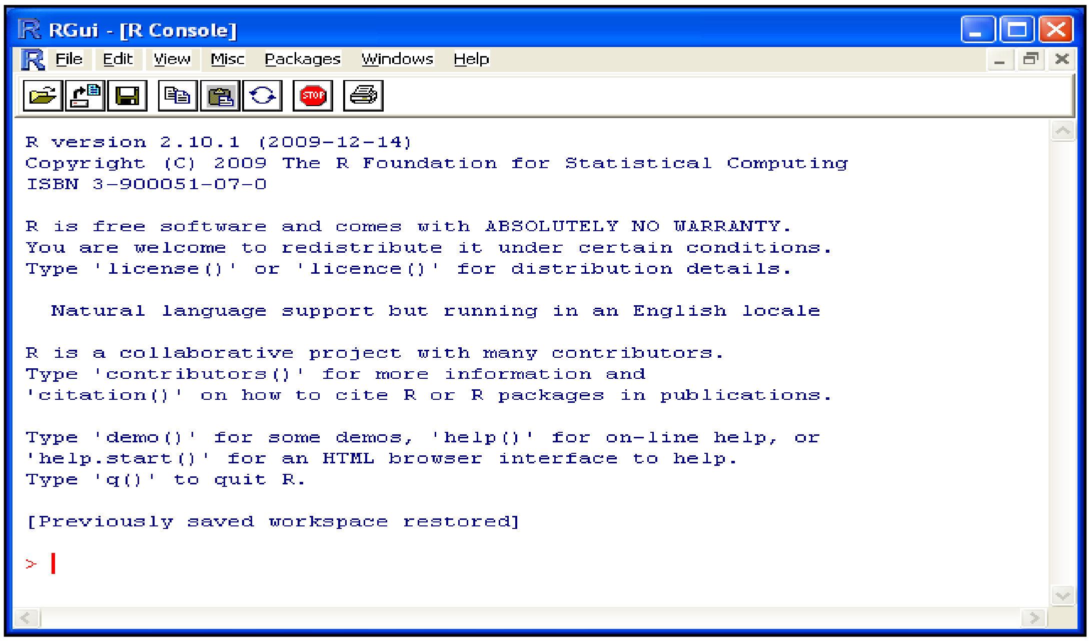
In version 8.0, two extensions named as Geostatistics and Spatial Analysis, are available. The tools in these extensions are related with spatial analysis but not SDA field. Until 2004, there is no SDA tool in ArcGIS Desktop. With the release of ArcGIS version 9.0 in May 2004, a new toolbar named as Spatial Statistics Analysis was added which has several SDA tools. Since then, ESRI has started to add SDA techniques in this toolbar. The progress of integrating SDA techniques in ArcInfo licensed ArcGIS Desktop version from 9.1 to 10.0 can be seen in Table 2.3. The total number of SDA tools is 10 in version 9.1 released in 2005, and 14 in version 10.0 released in 2010. The number of newly added SDA tools in five years is only 4. Similarly, the percentage of number of SDA tools over all GIS tools is very low and the ratio is almost constant in every version of ArcGIS Desktop. Grand total number of all tools in ArcInfo licensed ArcGIS Desktop is 954 in version 9.1 and 1387 in version 10.0. The percentage of SDA tools over all tools is only 1.05 in version 9.1 and 1.01 in version 10.0. These figures prove that the progress of ESRI Company for integrating SDA tools into their products is very slow and the full integration could not be achieved by the company in the near future, if the market has not boosted with an incremental demand from the GIS users.

Table 2. 3. The Number of SDA and GIS Tools in ArcGIS Desktop Version 9.1, 9.2, 9.3, 9.3.1 and 10.0 (ESRI ArcGIS Desktop Functionality Matrix 2010)

Spatial Statistics Analysis Toolbar in ESRI ArcGIS Desktop (ArcInfo Licensed)	**ArcGIS Desktop Versions and Their Release Date**				
	Ver. 9.1 in May,05	**Ver. 9.2 in Nov,06**	**Ver. 9.3 in June,08**	**Ver. 9.3.1 in May,09**	**Ver. 10.0 in June,10**
Number of SDA Tools	5	6	8	8	8
Number of Exploratory SDA (ESDA) Tools	5	5	5	5	6
Number of Utility Tools	8	9	12	12	12
Number of New Added Tools	-	2	5	0	1
Total Number of SDA Tools (including ESDA)	10	11	13	13	14
Total Number of Tools in the Toolbar	18	20	25	25	26
Grand Total Number of All Tools	954	1202	1238	?	1387
% (SDA+ESDA / Grand Total)	1.05%	0.92%	1.05%	-	1.01%

2.4.4. FLOSS Product

First reviewed FLOSS product related with SDA field is R, which is one of most widely used statistical package in the FLOSS environment (Figure 2.9). It is the FLOS version of S-plus. R is worldwide voluntarily supported FLOS software system (Chambers 2008). R is free and runs on almost any computer, including Windows, Macintosh, Linux and UNIX (Muenchen 2009). Venables et al. (2008) explain that R is mainly used for “data manipulation, calculation and graphical display”. Primarily, R is designed for scientific computing and graphics (Maindonald and Braun 2006). The origin of R has started with a research project in 1990s and has been a GNU project since 1997 (Chambers 2008).



```
R version 2.10.1 (2009-12-14)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> 
```

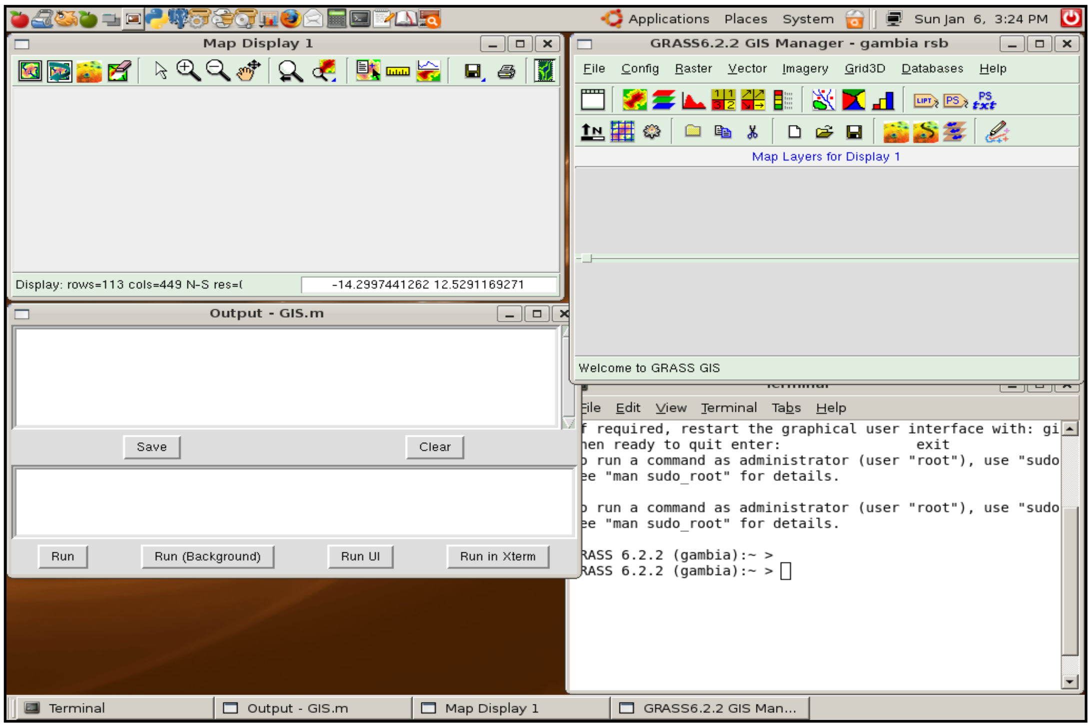
Figure 2. 9. Snapshot of R GUI

R is an environment and has a language with its own syntax. R language is adopted from S language, which is developed by AT&T Bell Laboratories in 1975. S language is the basis for the commercial S-PLUS, a statistical package software (Maindonald and Braun 2006). Chambers (2008) explains that R language remains “largely compatible with S language”. According to the award of John Chambers' 1998 Association for Computing Machinery Software, S language has altered how researchers analyze, visualize and manipulate data (Maindonald and Braun 2006). R reflects this philosophy and has evolved the S language (Chambers 2008). R Core group as the central control team handles the base system (Maindonald and Braun 2006). Many statistical techniques are supplied by packages (Venables et al. 2008). Thousand packages in the central repository of R project support the base system (Chambers 2008). The support of community is very huge and R has many different packages for variety of fields including SDA.

The main disadvantage of R is that the operations in the R environment are done with the command line interface which is not user-friendly environment. The interface requires sort of the knowledge of programming while using command prompt. This is not easy for the GIS users who have no programming or statistical knowledge. Learning R can be a difficult task, especially if the user does not have any programming experience. Another important disadvantage of R is that it is not a

GIS software product and has got a different mentality or style of handling spatial data that is too much complicated and different for many GIS users, who have get accustomed to work with user friendly menu driven GIS software.

Another reviewed FLOS desktop GIS software product is GRASS, which is the abbreviation of Geographic Resources Analysis Support System (Figure 2.10). The software is generally used for “data management, image processing, graphics production, spatial modeling, and visualization” of variety of spatial data formats (GRASS GIS 2008a). GRASS GIS manipulates raster and vector data, processes multi spectral image data, creates, manages, and stores spatial data (GRASS GIS 2008b). GRASS GIS have both windows interface and command line terminal. GRASS GIS supports a wide range of applications and used both in academic and commercial world (GRASS GIS 2008a).



```
f required, restart the graphical user interface with: gi
hen ready to quit enter:                          exit
o run a command as administrator (user "root"), use "sudo
ee "man sudo_root" for details.

o run a command as administrator (user "root"), use "sudo
ee "man sudo_root" for details.

RASS 6.2.2 (gambia):~ >
RASS 6.2.2 (gambia):~ >
```

Figure 2. 10. Snapshot of GRASS GIS in Ubuntu Operating System

GRASS GIS has started to be developed in a UNIX environment in 1982. The software is ported to many other systems like Windows and Mac OS X (GRASS GIS 2008b). At first, the software can run in Windows operating system by means of a Cygwin emulator, and then, the development for the native port of GRASS GIS for

Windows OS has started. Native windows port named as WinGRASS is in the experimental stage in August 2008. The software has unique properties in terms of portability, which makes GRASS GIS that can work under different OS (Neteler et. al 2008). GRASS GIS is written with ANSI-C programming language and has a preliminary C++ interface (GRASS GIS 2008a). GRASS GIS is integrated with GDAL and PROJ libraries (Neteler and Mitasova 2008). By this way, the software can support range of raster and vector formats, as well as projections.

The development of GRASS GIS has started in US Army Construction Engineering Research Laboratories in 1982 and continued until 1995 in this laboratory (GRASS GIS 2008a). The aim of development is to disseminate the usage of GIS in the land management and environmental planning. When ArcView GIS series 3 is released by ESRI in 1995, the development of GRASS GIS was terminated. Within a couple of years, the development was transformed into an FLOSS project by the University of Hannover. And since then, the project is managed by the international GRASS Development Team (GRASS GIS 2008c). The software was published as free software under GNU GPL in 1999 (Neteler et al. 2008).

The disadvantage of GRASS GIS is that the visual style of GRASS GIS interface and usage manner belong to old fashioned UNIX. Although there is a port of GRASS GIS to Windows OS, it is still in the experimental stage in August 2008. That kind of radical change in the structure of GRASS GIS makes any development effort very unstable for the external developers. This difficulty will continue until the software will have mature structure and reach to stable version. Another disadvantage of using this software is that it has different usage style and manner. For example, GRASS GIS requires learning new GIS data format. This format is a sort of database structure based on a group of files. The format requires getting familiar with new features such as “location” and “mapset”. Although GRASS GIS has a lot of GIS tools, the architectural structure of this software is very old. It is hoped that renewal structure of GRASS GIS is succeeded with the WinGRASS project.

CHAPTER 3

FREE/LIBRE AND OPEN SOURCE SOFTWARE (FLOSS)

3.1. Definition of FLOSS

The development of free/libre and open source software (FLOSS) is a new and different software production methodology and has started to be used with GNU/Linux Project. There is a slight difference between the term “free software” and “open source”. Free software is thought of most appropriately as freedom. Open Source definition is supported mainly in the developer community (Cascadoss 2007a).

Richard Stallman, founder of Free Software Foundation, mentions that there are four types of freedom to define the term "free software" (Free Software Foundation 2008):

> ***Freedom 0****: The freedom to run the program, for any purpose,*
> ***Freedom 1****: The freedom to study how the program works,*
> ***Freedom 2****: The freedom to redistribute copies so you can help your neighbor,*
> ***Freedom 3****: The freedom to distribute copies of your modified versions to others.*

Stallman (2002) explains that free software provides to their users “a freedom to run, copy, distribute, study, change and improve the software”. The software can be installed and run on unlimited number of computers through Freedom 0. Free Software Foundation underlines that license of the software produced or modified derivatively must be identical with license of previous software in order to maintain aforementioned freedoms. This necessity signifies surrender of copyright. Freedom 2 mentions that the software must clearly indicate license to users when it is distributed. Publishing of the software derived source code is a necessity, when it is distributed. Richard Stallman frequently uses his famous slogan; “think ‘free’ as in ‘free speech’ not as in ‘free beer’” in order to explain the meaning of free software (Chopra and Dexter 2008). In other words, “free as in free speech” refers to that there is no restriction in to use, modify and distribute the software. “Not as in free beer” refers to that a business model can be established by using free software products. The term “free” does not mean to no fee in the free software.

Proprietary software is restricted and protected from the users with the copyright law. Copyright sustains the privatization of the software. Stallman has made an analogy with the term “copyright” and produced “copyleft”. The term “copyleft” is known as providing source code so that users could modify, enhance, and customize their software without any restriction. This term also includes the distribution of a modified version with the source code (Moody 2002 cited in Chopra and Dexter 2008). Copyleft is the opposite meaning of copyright law. According to Stallman (2002), copyleft becomes a meaning of keeping the software free.

In 1998, a group of free software developers launched Open Source Initiative (Chopra and Dexter 2008). They have requested to gain a higher profile from free software movement. Open Source Initiative is an organization founded by Eric Raymond to promote open source software and development strategies (Sandred 2001). Open source is defined by Open Source Initiative according to 10 criteria which the distribution terms of open source software must comply (Open Source Initiative 2010a):

1. *Free redistribution*
2. *Including source code*
3. *Allowing modifications and derived works*
4. *Integrity of the author's source code*
5. *No discrimination against persons or groups*
6. *No discrimination against fields of endeavor*
7. *Distribution of license*
8. *License must not be specific to a product*
9. *License must not restrict other software*
10. *License must be technology-neutral*

There are nearly 70 open source licenses approved by Open Source Initiative (2010b). The licenses evolve in time and some of them have become superseded and retired. Imposed restrictions of open source licenses vary in different degrees. For example, Artistic license imposes few restrictions (Deitel et al. 2002).

Eric Raymond is a founder of Open Source Initiative, also a FLOSS developer and one of the first observers in the FLOSS political economy (Chopra and Dexter 2008). Eric Raymond's article "Cathedral and Bazaar" is regarded as the manifesto of the open source movement (Vries et. al 2008). The article compares two ways of development; cathedral (proprietary) and bazaar (open source) style development. Koch (2005) describes that little hierarchy exists in the bazaar style and development appears chaotic. Cathedral style development happens in isolation from users and with rigid authority from the top (Bates and Stone 2006). According to Raymond, Linux was the first large scale project that demonstrated the efficiency of the bazaar style (Söderberg 2008). Raymond (2001) defines that main difference between the cathedral and bazaar styles is that in the cathedral style, "the programming, bugs and development problems are tricky, insidious and deep phenomena". For this reason, upgrades of cathedral software must be realized a long period of testing to ensure that all bugs are removed from the program (Söderberg 2008). However, in the bazaar style, the bugs are generally accepted as normal phenomena and fixed in time (Raymond 2001). In this model, anyone with Internet access and programming skills can participate in the development process (Söderberg 2008). This is the advantageous of open source software over proprietary software. By this way, "the bugs are more rapidly detected and fixed" in the open source style development (Koch 2005). Raymond (2001) advices that release should be done often in open source software in order to get more corrections.

Free Software Foundation (2008) "recommends using the term 'free software' instead of 'open source'". Open source focuses on technical issues, but ignores the value of freedom. "Libre" is mostly used within the free software movement in order to emphasize the freedom. Whereas Open Source Initiative has preferred to declare a new word "open source" instead of using "free software" due to ambiguity meaning of the word "free" in English language. The Open Source Initiative claimed the term "free software" was too confusing for managers. For this reason, the initiative offered a new term "open source" to increase the attractiveness of the free software development model to the business community (Chopra and Dexter 2008). Stallman (2002) mentions that “open source is a development methodology, whereas free software is a social movement”.

The choice of using both terms “free software” and “open source” together in this thesis is done deliberately in order to avoid ambiguity meaning of “free”. The term “free” refers to “libre”, which is the freedom to run, copy, distribute, study, change and improve the source code of the software. Alternative term for “free software” and “open source” is to use "Free/Libre and Open Source Software (FLOSS)”. In this thesis, the term “FLOSS” is used to refer to both “free software” and “open source” which are accepted as in the same meaning.

Briefly, free software provides all users freedom to run, distribute, modify and redistribute copies of modified versions to others. The free software underlines to have a right to access source code, to modify the source code and also to redistribute the modified source code. Source code is a human-readable instruction determining how software would run and which tasks would do by the program in the computer. Krysa and Sedek (2008) define source code as the uncompiled, non-executable code of a computer program stored in source files. It is not possible to develop the software without accessing the source code. Any modifications could not be carried out on compiled binary code, but this could be done on the source code (Krysa and Sedek 2008). Therefore, access to the source code is a fundamental condition for the freedom to study how the software works and to redistribute the derived works.

In free software world, a person who knows how to develop a program has the freedom to develop new features in the software completely free of charge without facing any legal obstacle. If developer can use and improve codes written by others, then other developers should be able to use codes written by that developer. This is a necessary in the free software in order to maintain the continuity of the software development. This necessity is legally sustained with the GNU General Public License (GPL). When the software is licensed under the GNU GPL, the derived development work originated from that software should also be licensed under the GNU GPL (Chopra and Dexter 2008). This is the protection of GNU GPL for free software that is guaranteed legally with this license. This protection avoids free software to be used with the proprietary software. Although this kind of restriction limits the use of free software, it provides the guarantee of accessing to the source code of the software as well as modified versions of the original software. GNU GPL also gives a right to the programmer to develop the software without requesting any permission from the author (Söderberg 2008). Free software movement uses GNU GPL software licenses to control over the source code or in other words the knowledge of developing the software (Koch 2005). However, open source licenses do not oblige any license to attach the derivatives of the original code (Söderberg 2008). For example, Berkeley Standard Distribution license permits the creation of proprietary derivative works from open source software (Vries et. al 2008). Permitting to create derivative works mean to rip, mix, burn, and release the open source software under the development of proprietary software (Söderberg 2008). Vries et al. (2008) mention that the advantage of this license is to allow “maximum freedom in using the source code to create derivative works” within the closed source software. However, this kind of freedom could cause in danger of not sharing the source code of this derivate works with the public. This is the risk for continuity of software development from the point of view of free software.

Most of GPL products are distributed free, because it is difficult to sell the license of software whose source code is provided to be accessible for everyone through the Internet. Software can be charged as follows when considered from perspective of GPL definition; a charge can be requested when distributing copies of the software physically (CD, DVD media, etc.) as well as a charge might be demanded when giving guarantee for the software or to provide technical service to users.

The opposite of free software is proprietary software or closed source software. Source code of the proprietary software is distributed as binary. When the source code is compiled, it is converted into binary code. Binary distribution completely prevents for developers to access to the source code of the software. Also, unauthorized access the source code of the proprietary software is prohibited with both the copyright law and the license of the software. Even if a developer adds a feature to proprietary software, the developer would not have the right to redistribute this improvement legally. Proprietary software imposes many restrictions to the developers and users by means of license agreements. Since nobody except manufacturers of the software has the right or the authorization to examine how the program works, it is impossible for users to know exactly which tasks are performed in the computer. As a natural result of this prohibition, when an error is occurred, it is hard to guess the reason of this problem by the user. Only the developer of that software can know the reason of failure and explain the error how to fix it in the proprietary software. Fixing and developing the proprietary software are under the monopoly of manufacturer of that proprietary software (Dalgıç 2007). Free software world is the name of other world that does not include these restrictions. Moreover, the users do not have to pay high usage fees for each copy of the closed source program in this world.

There are many benefits to apply FLOSS development methodology for analyzing spatial data. According to the Bioconductor Project, which is an open source software development project for the analysis of biological data, the benefits of applying FLOSS development methodology are explained as (Bioconductor 2008):

- *Full access to algorithms and their implementation*
- *The ability to fix bugs, extend and improve the supplied software*
- *To encourage good scientific computing and statistical practice by providing appropriate tools and instructions*
- *To provide tools which allow researchers to explore and expand the methods used to analyze the data*
- *To ensure that the international scientific community is the owner of the software tools needed to carry out research*
- *To lead and encourage the commercial support for the development of tools*
- *To promote reproducible research by providing open and accessible tools that are carrying out scientific research*
- *To encourage users to become developers, either by contributing packages or preparing documentation*

The same benefits can be obtained also for SDA field. In order to gain these benefits, the FLOSS development methodology should be applied while developing specialized software for SDA field. Those benefits are gained with free software revolution. This revolution is shortly explained by Deitel et al. (2002) that the concept of free source technologies is not new. The development of free source technologies was an important factor in the growth of modern computing in 1960s. After the Internet was established, closed-source technologies and software became the norm in the software industry, and free source fell from popular use in the 1980s and early 1990s. In response to the closed nature of most commercial software and programmers' frustrations with the lack of responsiveness from closed source vendors, FLOS software regained popularity today.

Developing software is a right that should be shared with the public and everybody should have freedom to use this right in this environment. When developing new or derivative software, it is not required to develop everything from scratch. It should be possible to take advantage of software developed by others, to produce new software, to improve the software by adding new features and to redistribute the software for reuse by others. These possibilities exist with FLOSS development methodology. A right for the development of the software is protected by GNU GPL in this system. FLOSS ecosystem owes its presence to this concept.

3.2. Benefits of Using FLOS GIS Software for Utilization of GIS Field

Free/Libre and Open Source (FLOS) GIS software can increase the utilization of GIS field by providing cost reduction, supporting open standards, offering variety of products for free. The most important advantage of FLOSS compared to proprietary software is that a license fee payment is not required necessarily. FLOS GIS software eliminates the software license fee which is one of the factors preventing the spread of GIS world. Unlimited usage of FLOSS refers to two opportunities. First, the software can be used on any computer for any private or commercial purpose. Second, the software can be installed and run on unlimited number of computers. When the use of FLOS GIS software has increased, GIS field can become more widespread. In addition to this benefit for utilization, all these products support the open standards. Open standards are the best way to cover different spatial data formats and great variety of users. Also, complete utilization of GIS field in any organization requires different type of applications like web, desktop, database and mobile applications. This kind of variety increases total cost of the whole system, especially when it is requested to pay licensee fee for each copy of the application. FLOSS products can provide great amount of cost reduction for enterprise GIS. All FLOSS GIS products can be developed according to the demand of the organization, because source code of the product can be accessible from the Internet. Source codes are the valuable information resources for the developers to learn how the application works.

There are several key factors affecting the spread of FLOS GIS software such as which GIS software learned in universities, taught in hands-on GIS trainings and used by researchers on their studies. Manufacturers of proprietary GIS software offer their GIS products to be used in the universities of their homelands free of charge or for a very low fee in order to popularize their GIS products. However, the same charging policy is not always applied to the countries where the software is exported. This situation limits the use of GIS application especially in the public sector. Public organizations can buy only a few licenses which decrease product variety or completely prevent utilization of this technology. GIS experts cannot find a chance to access and learn different variety of products due to limited number of licenses or they are pushed to commit crime by using illegal copies. High license fee prevents the spread of GIS in the public and especially in the private sector. While relative discounts are available for public sector and universities, there is not such a price regulation for private sector. This situation causes an unavoidable dependency on government in terms of dissemination of GIS field.

Spread of FLOSS should first start in universities in order to utilize the GIS field. FLOSS world provides great opportunities to researchers. FLOSS saves researchers from reproducing an algorithm that was already applied. FLOSS is a great information resource for researchers working on GIS field to learn how SDA technique works. Moreover, researcher using FLOSS owns a testing environment in which they can try new techniques and develop new techniques. FLOSS has a comprehensive documentation for researchers. The best source explaining how algorithm works is the source code of the software. Source code of FLOSS can be examined by everyone freely. FLOSS can be extended and modified in accordance with requirements of the research without any limitation. New developments and techniques invented in FLOSS can be published and shared with other researchers instantly. Reproducible results can be obtained when the researchers have a chance to setup the same software environment. This is the basic requirement in the discovery of new scientific methods; accessing to the same environment and having an opportunity to develop the techniques by applying with different algorithms.

Second way of utilization of GIS by the FLOSS products is to use open standards. Open standards are the best way to support every GIS user and data formats. Most of the FLOS GIS software supports open standards and open geographical data formats described by Open Source Geospatial Consortium. When FLOSS products supporting open standards are chosen to build national or regional spatial data infrastructures, the advantages provided by these products to GIS world are much more than those of proprietary products. Aim of establishing such a system is to access all GIS users and integrate many GIS data formats into one common platform. In order to reach this goal, spatial data infrastructure system must support open standards, commonly used GIS data formats and provide various solutions to convert to different GIS data formats. Nearly all of the FLOS GIS software is developed to support according to the specifications of Open Source Geospatial Consortium.

The utilization of GIS can be divided into four groups according to the use of GIS applications in the public organization in Turkey. However, there is no clear difference among the groups; some of listed features can take place in other groups as well. The purpose of grouping the utilization of GIS is to show which level of use can be reached when product variety is achieved in an organization.

The use of GIS in the first group is at the most basic level. This group represents that one or several personnel carry out GIS works in one unit of the organization. These users perform the required task by using GIS software and generally GIS data are stored in personal computers. In this case, the use of GIS remains quite limited within the organization. This group needs only desktop GIS software as minimum.

In the second group, a department is responsible for performing GIS related tasks in the organization. GIS department serves to meet requirements of other units within the organization. GIS data are stored on a server and shared within the department by a central and single database. Second group represents transition between personnel usage to enterprise GIS. It is possible to establish the aforementioned unit in charge, since the usage level of GIS in this group is higher than the one in the first group in terms of having several GIS software licenses and variety of GIS related products. This group requires desktop GIS software plus database application and interface for converting and storing spatial data in the database application.

In the third group, the use of GIS is not limited with one responsible department. One or more personnel in different departments can use GIS software and access geographical data. GIS users in different departments can access geographical data stored in multiple spatial databases distributed within several servers. GIS can be used by several units instead of a single responsible unit. In this group the use of GIS is spread to most of organization. Personnel can quickly access to geographic data via Intranet, Internet and perform spatial analyses. GIS software is customized according to requirements of the organization. GIS is integrated with other enterprise systems like electronic document management system, enterprise resource planning and customer relationship management. Also customers can access to geographic data via Internet. GIS works like decision support system for the managers in the organization. This group requires variety of GIS applications like desktop GIS software, database, web mapping application, customized user interface and probably mobile GIS application.

Fourth and also the last group represents the most advanced GIS use. In this group, GIS is not a tool but a goal. This group uses GIS to conduct spatial research and development projects, to prepare and to provide national or regional geographic data infrastructures, to explore new analysis method, to develop new GIS software, to discover spatial technologies, to test new GIS applications. Some of the GIS usage listed for this group can only be achieved through studies performed in the universities. This group also contains the private commercial companies which perform to sell GIS products, to provide technical support about their products, to develop new GIS software, to make localization, to translate the manual and GUI, to customize the software by developing improvement such as add-on, plug-in, extension, custom toolbar, etc.

In order to utilize GIS more functional as in the second group in the public organizations in Turkey, an organization should have a complete information technologies infrastructure like server, personnel computers connected with each other via network, fast and stable Intranet and internet connection, database application, qualified personnel. Organization could have difficulty in reaching to this GIS level due to several reasons. License fee of proprietary GIS software can be very expensive. For example, one concurrent license of ESRI ArcGIS Desktop GIS program licensed with ArcInfo functionality is $30,000 plus value added tax in 2010 in Turkey. However, FLOS GIS software can dramatically lower the prices. The closed source software license fee is one of the drawbacks in the spread of GIS in Turkey. There can be other reasons which prevent the utilization of GIS in the public organization. The public organization has not yet setup a network and could not have enough number of personnel computers and not have any powerful servers. The organization does not have specific hardware like plotter, scanner and GPS which are required while creating the geographical data. Also organization could have difficulty in employing qualified GIS experts and developers.

Third way of utilization of GIS by the FLOSS products is to develop enterprise GIS. Enterprise GIS can be utilized in the organization when several software products are owned by the organization. Enterprise GIS contains web mapping applications, desktop software, database application, interface converting spatial data into the database, development libraries and applications developed for specialized fields like 3D analyses, spatial modeling, network analysis and artificial intelligence. These applications could not be provided within single product up to now. Some of aforementioned applications were included within one product and some are marketed as a plug-in or a standalone product. This means to pay many licensee fees for variety of products in order to setup enterprise GIS in the organization. Also license fee for each copy of desktop GIS software should be paid to the GIS vendor. GIS software is released generally in every two years, which means that license fee should be renewed in every two years. This is the total cost of owning proprietary enterprise GIS software. When all these costs are taken into account, it is obviously seen that one of the most important factors preventing to facilitate enterprise GIS in the organization is software license fees itself.

Although it is considered that the abilities of FLOS GIS software are relatively limited in certain subjects, cost advantageous of owning FLOS GIS software as enterprise GIS is significant. Moreover, missing features can be developed by programmer or it can be requested from the community developer. Cost of adding needed features to FLOS GIS software and total cost of proprietary GIS software should be compared before deciding which software to use. Proprietary GIS software meets general demands of users. Proprietary GIS software should also be customized and adopted in order to do specific tasks in accordance with the demands of the organization. The development work should also be carried out in order to use the proprietary GIS software as enterprise system. This situation shows that development work is required for both proprietary and FLOS GIS software as well. It is not possible to avoid development cost for enterprise GIS whether the software is a proprietary or free.

When FLOSS products are preferred for setting up enterprise GIS, it is possible to access source code permanently and this situation prevents any monopolization in regard to continuity of development work. A programmer with experience of developing in the same programming language can continue to carry out the maintenance work. FLOSS product eliminates the necessity of having technical support from the manufacturer of the software. Enterprise systems should continuously be developed to adopt the changes that can occur in the structure of the organization. Enterprise application, which is not adopted according to new needs, becomes unusable over time, lose reliability and after a while they become completely useless. They have become to be replaced by a new system partially or completely. The most important advantage of system development with FLOSS is that the system can continuously be developed by the programmer and new features developed by the FLOS community can be integrated with the implemented system in the organization, because there is no obstacle to access to the source code of newly developed part of the system.

Software products to establish enterprise GIS application can be obtained from FLOSS world. Different programs are required to establish enterprise GIS application from different information technology fields like web server, internet security applications, database application, web and desktop GIS software. These products can be found in FLOSS world and configured to run together. The cost of establishing enterprise system in terms of software license fee can be eliminated when FLOSS products are used.

Enterprise GIS application can be realized by using FLOSS products. In this case, one of the FLOS Web GIS software publishing geographical data within web browser over a network such as internet or Intranet through one or more servers integrated with database should be preferred to realize this task. Several FLOSS products should be installed on the server and configured to run together. Necessary configuration settings should be carried out. This can be difficult for GIS expert to setup the environment such as installation of additional plug-in needed to provide extra features to the software, setting up user’s security configuration in the database and establishing GIS interfaces which provides the storage of spatial data in the database, etc.

There are several FLOSS products that can be used to implement as enterprise GIS. In order to decide which FLOSS products would be used, the product should be evaluated according to the several selection criteria. Quality of documentation, features provided by software product, compatibility to run with other products, supporting status for open standards, performance of the software for large size GIS data are very important subjects when deciding about which GIS product should be chosen to be implemented in the enterprise GIS. FLOSS project should be mature, supported by a broadly-participated community and have up-to-date configuration and compilation instructions explained properly. It will be better to learn installation and configuration tasks before deciding to use the software. If these tasks could not be carried out correctly, it would require asking a professional technical support from the community.

In FLOSS world, a lot of developed products are available for different fields. The most important subject to be searched while deciding which GIS product is used, is to look at whether the project is supported by a mature community and have sufficient documentation or not. Moreover, FLOSS projects can also be supported by private commercial companies. Technical service can be acquired from these companies. To pay technical service for FLOSS can be meaningless for managers in software environment where most products are free of charge. Requesting technical service from the community with fee speeds up the development work for your organization. Most appropriate way to benefit from FLOSS projects for any organization is to be a sponsor or donors to the project. This will make many development tasks easy and create strong communication between community developer and the programmer of the organization.

Establishing enterprise GIS in the organization is challenging task. Enterprise GIS requires harmonizing several application areas like web, GIS, database and even mobile products. The variety of products causes to increase total cost of proprietary licensee fee and the amount of development work carried out. FLOSS can sustain huge cost reduction in terms of licensee fee. This provides great flexibility for managers in the organization to support the development cost of the system. Development work should be carried out both for proprietary products and FLOSS.

This necessity is unavoidable part for the establishment of enterprise GIS. When it is decided to use FLOSS to setup enterprise GIS, it could be important to take technical support from the community developers. This kind of support would determine the success of the system and cuts the spent time for the completion of the system.

FLOS GIS software can not be enough mature as compared to proprietary GIS products in some parts, but if FLOSS are supported to develop more, there is no reason to have full functional program like proprietary products, because they are open to be developed by everyone freely.

3.3. Basics of FLOSS Development

The most important advantage brought by FLOSS world is to provide the capability and possibility of taking intellectual and proprietary rights of software from a single corporation and sharing these rights with the public. FLOSS provides the freedom to develop software over again in any way to all people. The aim of FLOSS is not to consume but to produce. FLOSS philosophy emphasizes on several freedoms such as a right to access, to modify and to redistribute the source code, and organizes how these freedoms can be used as a social production tool.

Other advantages and freedoms brought by FLOSS are possibility and opportunity of reducing costs, improving the software in regard to personal needs or asking a programmer to do this economically and quickly instead of you, understanding whether the software is safe by studying how it runs, having a secure software since it was tested and studied by many programmers, receiving updates and new features much more faster due to their delivery over Internet freely. Also, if users prefer to use FLOSS products on Linux instead of proprietary operating systems, then they do not have to challenge computer viruses as well. This provides an extra safety environment. The greatest benefit of FLOSS for the users is to have the software free of charge or they demand a much lower charge compared to proprietary software.

Accessing to the development libraries and tools in FLOSS environment is relatively much easier and cheaper as compared with proprietary software environment. Cost advantages of FLOSS against proprietary software should be considered with reference to “Total Cost of Ownership” concept. According to a research conducted in 2001 by Financial Audit Bureau of Baviera in Germany, FLOSS has lower total cost of ownership and a higher investment return compared to their alternatives. This report underlines that cost comparisons between free and proprietary software is much more beyond license fees. Also, it emphasizes that indirect costs (hardware upgrades, access to data in old format) have to be considered besides direct costs (license fees, installation cost, training and support).

Another study about the total cost of ownership values have been carried out by Cybersource Company (Gözükeleş 2004). In this study, a business enterprise has been modeled by considering criteria such as personnel with 250 computers, servers and workstations with Internet connection, e-business system, network wiring and equipment, standard software, wages of information technology specialists required to maintain continuity of infrastructure. This model has been developed through two alternatives. In first alternative, it was assumed that the enterprise already had an existing system. In second alternative, a starting from scratch approach has been used. A comparative result of study conducted by Cybersource Company is shown in Table 3.1 (Cybersource 2004). Total cost of ownership of FLOSS is 36% cheaper in the first alternative and 26% cheaper in the second one with respect to proprietary software. This model has been calculated as per 250 computers only. Savings would be even more striking if hundreds of thousands of computers are considered.

Table 3. 1. Comparison of Total Cost of Ownership (Cybersource 2004)

Total Cost of Ownership for Three Years	**Microsoft Solution**	**Linux Solution**	**Savings by Open Source**	**Savings as percentage**
Use of existing hardware equipment and infrastructure	$1,066,712	$682,090	$384,622	% 36
New hardware equipment and infrastructure is bought	$1,366,883	$1,012,260	$354,623	% 26

FLOSS is a continuously expanding information source for those who know how to use this source. Today all legal and technical means required to develop software are available in FLOSS development environment, as in the past. Developers who are in capable of writing code collectively and voluntarily are always in demand. By means of technical tools provided by FLOSS, code writing can be carried out collectively independent of time and location.

The possibility of having a right to use development tools makes FLOSS development technically easier. In order to protect these freedoms, philosophy for developing FLOSS should be understood and owned by the users. This philosophy can be supported in the simplest term by using FLOSS and contributing to the FLOSS projects. In order to benefit from these utilities, the rules of FLOSS world must be perceived. Support from developers and users should become widespread in order to carry out FLOSS development. Motivation of members of community depends on contributions received from the users. Difficulties related with developing the software continuously can be overcome only when the members are satisfied economically and socially.

Software needs to be developed continuously to sustain its presence. Each program needs continuous maintenance due to improving performance, major changes in the dependent library or in the operating system and implementing possible new standards and data formats, etc. The cost of showing initial efforts to develop the software, keeping continuously to maintain the software, spending efforts to make the software free of bug and fixing security vulnerabilities, adding new features, briefly improving the software can be overcome by applying FLOSS development methodology. These costs can be eliminated when the project has become well known, their products are widely used, and the project is supported by active and rapid growing community. This progress can be realized when more users learn their rights, own the FLOSS philosophy, contribute the FLOSS projects and prefer to use more FLOSS products in their daily works instead of using proprietary software.

FLOSS products gain more importance in many different fields by offering similar service quality as proprietary products can do. Today, some FLOSS products compete against proprietary products. Developers and users have great responsibilities for the development and spread of FLOSS products. Hopefully, many developers will start to contribute to develop more FLOSS project and users will prefer to use more frequently FLOSS products due to freedoms and social benefits provided by FLOSS.

3.3.1. Continuity of FLOSS Project

A FLOSS project should have an active community in order to continue to maintain the development of their product. In the project, project governance structure should be setup and shared with the users. The project governance structure includes a mission statement, a conflict resolution procedure, and a procedure that explains how new developer can be accepted as a member in the core development team (Cascadoss 2007a). For example; QGIS Project is organized under these responsibilities; packaging team leader and members, code maintainers, manual team leader and contributors, manual and GUI translator team leader and members, financial advisor and assistants, release manager and assistants (Quantum GIS 2010b). Newcomers should contribute to the project free of charge according to their talents. Contribution of newcomers can be in many different ways such as code development, contributing the documentation, reporting bug, advertising the product and the project, donating, translation, etc. The aim of organizing the members of the community is to gain these contributions coming from new users and adopt them to be a new member of the community.

The responsibilities of members of this community should be organized by either the originator/leader of the project, the core development group or the most experienced developer in the project. The decisions can be discussed via e-mail and are taken by consensus of the members of the group. Most FLOSS projects leadership is meritocratic in nature (Cascadoss 2007a). This means that the member, who has made more valuable contributions to the project, has a more right to advice strategic decisions. Strategic decisions are taken according to the consensus between the members of the community. Consensus is vital for the management of FLOSS projects, because there is a potential risk to occur a fork. Fogel (2006) defines the term “fork” as anyone can take a copy of the source code and start a competing project. Fork is a bad thing in the project and tried to avoid it by organizing the project as democratic way, because the available developer’s manpower is limited and with the fork, one group of developers and users are lost from the original project. Fogel (2006) adds that the possibility of being a fork is the main reason for not being true dictators in FLOSS projects.

Continuing the development of FLOSS product depends on the motivation of the members of the community. In order to satisfy the members economically, the project should be managed to gain the financial support from users, private commercial companies and public organizations. The most logical way to get financial support while developing FLOSS products is to provide technical services to customers. Although there are not enough corporate businesses providing technical support service for FLOS GIS software in Turkey, it is thought that technical service for FLOS GIS software will increase and become more widespread due to benefits provided by FLOSS development method.

A counter thought in this regard states that technical support services cannot be provided by corporate businesses. This view claims that this technical support service should continue to be provided by small companies which employ a few people who are talent in the development of free software. The release of new version of FLOSS products in active projects is much faster compared to proprietary software products. This speed can also be seen while fixing the errors. Therefore, it is considered that any corporate business cannot provide technical support as cost effective as a small company. Corporate businesses cannot compete against originality and innovation of technology that small companies can produce in FLOSS world. Adaptability skill, speed and maneuverability of small companies are much higher than large corporations. This is the power of small companies provided with the use of FLOSS development tools.

The power of large corporations comes from their wide range of customers. Presenting the software as a brand in the market requires an investment apart from software development area which makes small companies dependent on large corporations. The cost for developing and maintaining continuously the software and advertising can be hard to overcome especially until the project has become popular. However, when FLOSS development methodology is applied, FLOSS allows the widest possible distribution of a software package at very low cost with minimal marketing (Cascadoss 2007a). There is always a keen competition between small companies and large corporations in regard to providing technical support service and marketing the technology. When considered from this point of view, both situations can be possible in future. It seems that technical service for FLOS GIS software continues to be provided both small companies and large corporations.

In addition to the provision of technical service, the project can be managed on the based on different business models. FLOSS development methodology has created several business models observed in the market and identified in the literature by Cascadoss (2007a) Project as follows;

- **The Dual Licensing Model:** Companies make their products with both free and commercial license like Qt and PyQt.
- **The Support Seller Model:** Companies that provide paid support for FLOSS products they developed.
- **The Third Party Support Seller Model:** Similar model with the support seller but the paid support is provided by the third party companies.
- **The Platform Provider Model:** Companies bundle several FLOSS products into a complete platform and guarantee the quality of the integrated platform.
- **The Consulting Model:** Companies provide consultancy with respect to FLOSS products based on knowledge or experience of working with FLOSS.
- **The Software as a Service Model:** FLOSS is used to provide access to revenue generating online services.
- **The Added Value Provider Model:** Companies that create proprietary software derived from FLOSS.
- **The Accessorizing Model:** Selling services or physical items related to FLOSS such as books, hardware, etc.
- **The Loss Leader Model:** FLOSS product is not charged and used to market a proprietary product which offers more functionality.
- **The Widget Frosting Model:** Selling a combined offering of high value software and/or hardware components together with low cost FLOSS offerings. For example, selling Oracle Database application bundled with Linux operating system.

In order to continue to maintain the software, economical support can directly be taken from public organization with government policies. For example, China, Germany and Brazil adopt and support FLOSS as part of their national policies. Most European countries especially Germany, France and Spain have grown their own communities (Gonzalez-Barahona and Robles 2006). The contributions of Europe to the history of FLOSS are extensive. For example; the kernel of Linux is developed by Linus Torvalds, Finnish; Python by Guido van Rossum, Dutch; MySQL by Michael Monty Widenius, Swedish; PHP by Rasmus Lerdorf, Danish; KDE by Matthias Ettrich, German (Gonzalez-Barahona and Robles 2006).

In Turkey, FLOSS is not supported with national policies. Pardus project is one of the unique examples in Turkey that is developing with the support of one of public institution. If Pardus could become widespread among public bodies in future, this can lead to initiate many developments in the FLOSS projects. Pardus has been developing since 2004 in the TUBITAK National Electronics and Cryptology Research Institute. Pardus is one of Linux distribution licensed under GNU GPL and it can be downloaded from project official web site and installed for free of charge. The first release of this operating system was in December, 2005 (UEKAE 2010a). Names and release dates of Pardus versions are listed below (UEKAE 2010b):

1. *Pardus 1.0, December 2005,*
2. *Pardus 2007, December 2006,*
3. *Pardus 2007.1 FetisChaus, March 2007,*
4. *Pardus 2007.2 Caracal, July 2007,*
5. *Pardus 2007.3 Lynx, November 2007,*
6. *Pardus 2008, June 2008,*
7. *Pardus 2008.1 Hyaena, September 2008,*
8. *Pardus 2008.2 Canis aureus, January 2009,*
9. *Pardus 2009, July 2009,*
10. *Pardus 2009.1 Anthropoides virgo, January 2010,*
11. *Pardus 2009.2 Geronticus eremite, June 2010,*
12. *Pardus 2011-RC, December 2010.*

National Linux distribution, Pardus is a leading project which has started in FLOSS development environment with the support of public institutions. Pardus has started to become a known operating system not only in Turkey but also throughout the world. DistroWatch.com is web portal listing statistics for usage frequency of Linux distributions, and in every month several new distributions are added to this list. In August 2010, Pardus Linux was at rank 66 in the last one month, 41 in the last 3 months, 53 in the last 6 months, 41 in the last 12 months among 320 Linux distributions (Distrowatch 2010). It is an important success to be in the first 50 among all distributions of the last year.

Gelecek and Truva Linux Distributions are other two FLOS distribution examples that are still being developed in August 2010. Gelecek Distribution is developed by Gelecek Company and serves with two separate products named Desktop Vision and Enterprise Linux. Truva Distribution is a local Linux distribution based on

Slackware since 2004 and version 2.0 release candidate has been released in March 2008. In DistroWatch.com, Gelecek and Truva distributions also take place in the list (Distrowatch 2008). Several Linux distribution projects like Turkuaz, Turanid and Turkix distributions in Turkey have been terminated as it happened in the world. Projects like these and similar ones are terminated due to insufficient support and lack of broad community participation. Contributions and supports are the most crucial factors determining the continuity of FLOSS projects.

The support can come from public organization, private company and non-governmental organization. The support of users and developers can be in the form of making volunteer contribution to the project. Gözükeleş (2006) explains the reasons of volunteer contributions of Turkish free software developers (Table 3.2). According to this study, the main reason of Turkish free software developers making volunteer contribution is the information share, cooperation and will to contribute for a better world. Minority of Turkish free software developers' motivation reason is to develop the business opportunities. It seems that Turkish free software developers do not concern the business model of FLOSS development methodology.

Table 3. 2. Motivation Reasons of Turkish Free Software Developers (Gözükeleş 2006)

Motivation Reasons	**Percentage (%)**
Information share, cooperation and will to contribute for a better world	67.2
To spend good time during programming	21.8
To learn new skills and self development	10.9
Due to thought that software is not proprietary	7.8
By nationalist feelings	4.7
To solve a problem that cannot be solved by proprietary software	4.7
To develop business opportunities	1.6

When its social and economic dimensions are considered, maintaining to develop a FLOSS project is a challenging and laboring work. Major challenge for developing a FLOSS project is to maintain code-writing with patience and discipline continuously. In Turkey, many FLOSS projects ended without reaching a certain maturity due to insufficient interest of users and not achieving a community around the project. Starting and developing a FLOSS project requires a very long time period. It is estimated that it can take 5 to 10 years depending on conditions, situation and software in question. It cannot be possible to create a community supporting a FLOSS project within a time period shorter than similar durations.

Software development requires qualified human resource. Contribution of developers to FLOSS project as qualified human resources could be very low in the software industry. Developing FLOSS and creating a community around the FLOSS project is a very labor intensive work which requires too much time. Thus it cannot be possible for this qualified human resource to offer their services free of charge for a long time. This situation is one of the difficulties for FLOSS project to develop mature software as full featured program. It is considered that financial support especially at the beginning phase of the project coming from a public organization, university, government or private company will be generally inevitable to maintain FLOSS projects.

The exact number and qualifications of free software developers in Turkey are not known. A study that can be connected to this subject was conducted by International Institute of Infonomics in 2002 (Infonomics 2007). Findings of this study's final report on FLOSS are based on results of a survey which involves approximately 500 free software developers. This Internet survey was announced through well-known internet portals and e-mail lists in FLOSS world. Aim of the study is to collect basic data which will determine importance and role of FLOSS in current economy. According to this report, the percentage of Turkish free software developers participated in this survey is only 0.4. Distribution of numbers of programmers by nationality is shown in Figure 3.1. While 71% of free software developers in survey are from Europe, 13% is from North America and remaining 16% is from other countries. Turkey is at third rank from the last after Luxembourg and Greece respectively. A similar rank appears for Turkey in regard to distribution of locations where these programmers live or work (Figure 3.2). The percentage of developer living or working in Turkey is 0.3. Similarly, Turkey is at third rank from the last after Luxembourg and Greece respectively.

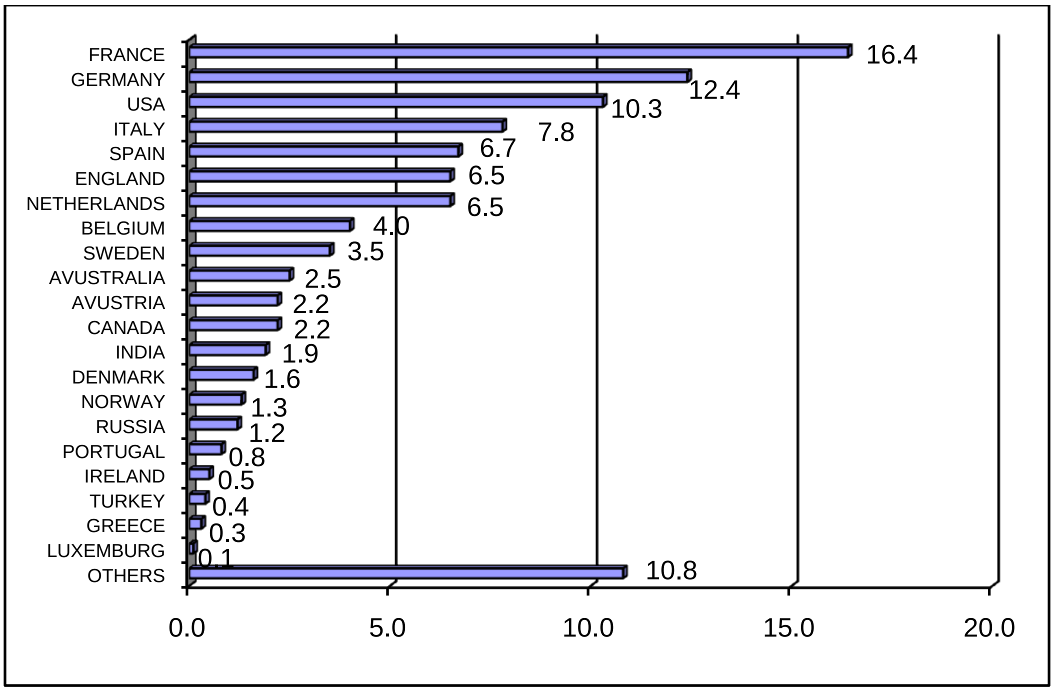


Figure 3. 1. Distribution of Free Software Developers by Nationality (%) (Infonomics 2007)

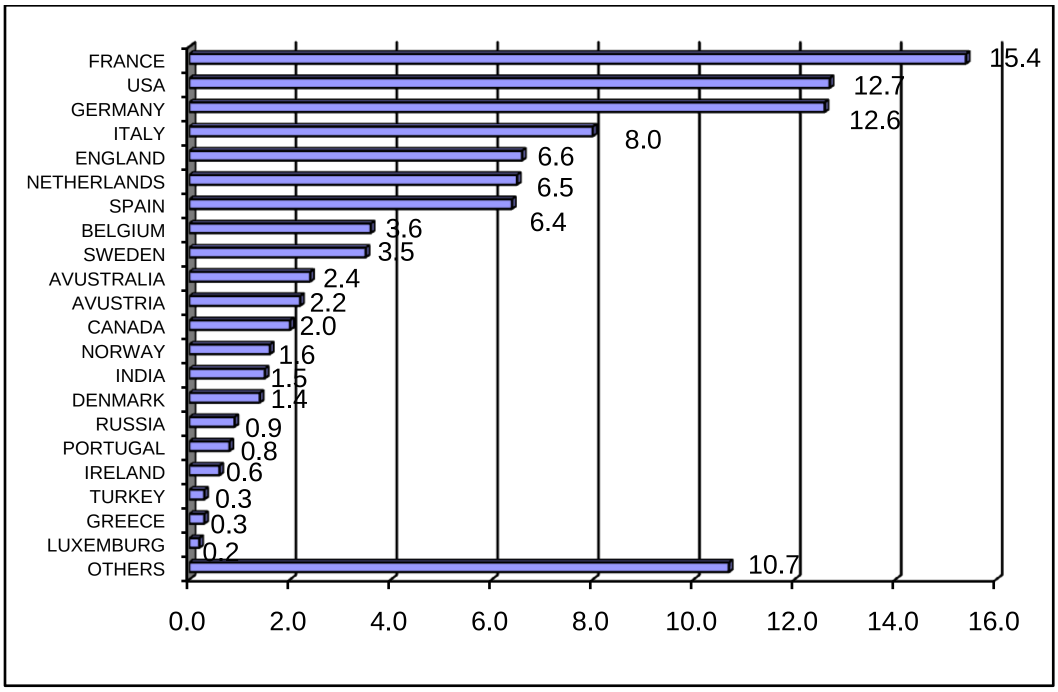


Figure 3. 2. Distribution of Free Software Developers by Countries Where They Live or Work (%) (Infonomics 2007)

Distribution of more preferred countries of free software developers for working purpose and their mobility patterns were determined through information on their nationalities and countries they work in (Infonomics 2007). This pattern was found out by comparison of Figure 3.1 and Figure 3.2. There is not a big difference between the countries where free software developers work and their nationalities. The percentage of free software developers working in a foreign country is 10. While Germany and United States replaces each other's rank, Spain and Russian Federation are less preferred by free software developers to work in relatively (Figure 3.3). While United States is the most preferred country as seen in distribution of more preferred countries to work in, France is the least preferred country by free software developers.

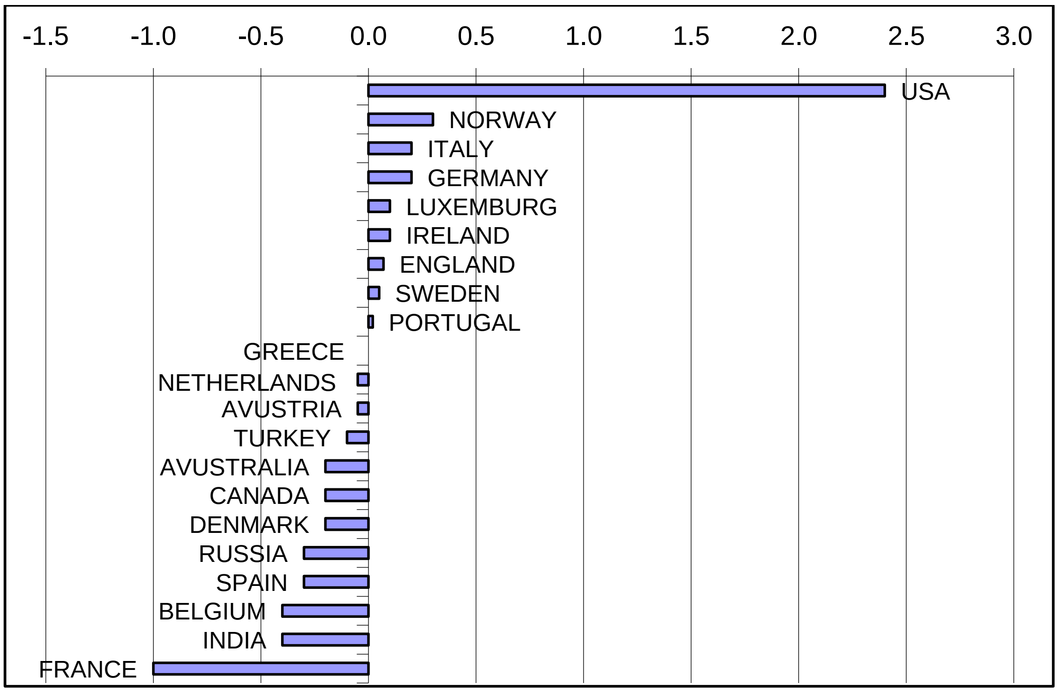


Figure 3. 3. Distribution of Mostly Preferred Countries by Free Software Developers to Work in (Infonomics 2007)

Turkey is at fourth rank in less preferred countries after Greece, Netherlands and Austria. Any European country was not as attractive as United States in terms of more preferred country by free software developers to work in. The reason for this result is unknown since related questions were not included in the survey (Infonomics 2007). As it can be seen from the figures, the number of free software developer in Turkey is low and Turkey is the least preferred country to work or live by the free software developers.

3.3.2. Developments in the FLOSS Project

Development in the FLOSS project can continue with an active community. The failure for creating a community around the project can terminate the FLOSS project. Creating a community requires a different management approach rather than programming skills. Management of this community composed of users and volunteer contributors generally requires extra time and effort. Software development environment should be designed in such a way that when a user visits the environment for the first time, the user can find a path to help the project. All software development processes should be observable by users. Software development decisions should be open and accessible. For example, if the developer member of the community uses communication (chat) programs to take strategic decisions about new direction of software development, written messages should be recorded and shared with everybody by publishing through the Internet.

There can be many casual reasons preventing formation of a community. Software without users cannot survive for a long time. If the software or development environment draws attention of users, then the project can gain contribution. Therefore, instructions on how to contribute should be prepared for new comers. By this way, it is possible to create a community around the software. This also creates a decrease in the work load of developers by transferring routine part of development works to members of the community. If a developer has more free time, more features could be possibly added to the software.

FLOSS world grows by both voluntary and commercial contributions, and enlarges more day by day. This economy always requires specialists and voluntary contributions. Preparing reference manuals, tutorials, user guides, frequently asked questions, providing end-user support by answering the questions of users in the e-mail lists and forum, advertising the program to be used more widely, testing the program, reporting bug, preparing packages, translating the interface and documentation into his native language, announcing news about the project and advertising the developed product are considered as some of tasks that should be carried out in the project by voluntary contributions that members of the community can provide for free. Each task requires specialized talent. For example, the person, who prepares the manual, is mostly native speaker, knows the features of the program and how to use the program very well. In order to gain these contributions, version control system, error tracking and reporting system, communication channels, test tools, package management systems, document management, debugging, memory error management, forums, and blogs are some of the features that are required to be implemented in the FLOSS project. FLOSS development methodology can be established by means of these features which can be provided with FLOSS products. For example, Subversion and Concurrent Versions System are two FLOSS product examples that are used for setting up version control system. Existence of these features maintains continuity of FLOSS project and helps the project to survive.

Each user can become a good contributor and take active role in the development of project. The user can contribute to the project in several ways; preparation of documentation, packaging, contributing to localization and translation tasks, program testing and error reporting, directly donating some money to the project and requesting new features. The purpose of these contributions can be a member of community, to educate himself, to apply academic study, or to learn the software development, etc. Contributions can return as reputation and prestige at first, and then can be financial gain in this sector. Prestige and possible financial gain are related to amount of contribution that is done.

Another development task in the FLOSS project is to setup e-mails as a searchable e-mail list. Members help between each other mainly by using e-mails in the project. The most essential and new information about the development of software can be found in the e-mails. New users should be able to search the e-mails in order to find a particular topic that has been discussed among the members in the past (Behlendorf 1999). Many strategic decisions about the development issues are taken by communicating through either e-mails or using chat programs between the community members. All those e-mails and the logs of chats should be shared with the public and can be accessible through web. Therefore, e-mails are archived as a researchable list in the FLOSS projects, and several e-mail lists are setup in order to separate each discussion topic. Particular topics are discussed in specific e-mail lists. When the user has subscribed to these e-mail lists, the user can learn and follow the development of project in his/her preferred topic.

Many FLOSS projects have valuable but messy documentation and spread all over the web. The documentation can be accessible from the web, although some of them become out dated and some of them are not documented. In the FLOSS environment, when the project is very active like QGIS, releasing of new version is very fast. For example, QGIS community released version from 0.9.1 to 0.9.2 and finally to 0.10.0 nearly in one year. In addition, each version has several release candidates. For this reason, the documentation could not reach the speed of development. Developers are very busy to develop the software and they are reluctant to write the documentation. Volunteer users contribute to the documentation when they learned the new feature while using the software.

Both the users and the developers can access to code repositories through Internet connection. But, only community developers have write permission to those repositories. The term “developer/programmer” and “community developer” are clearly different from each other in terms of works they do. It is important to reveal this difference in terms of expansion and development of FLOSS. Programmer researches FLOSS and uses the product which provides the best solution for his/her clients’ requests and s/he develops the software to add new features upon requested demands. Programmer delivers the fastest and most efficient solutions to his/her client. Community developer, on the other hand, commits himself to develop the software to be a product, continuously develops the software and publishes new versions rather than benefiting from it. Community developer can be described as a permanent member made him accepted among community and has authorization to write to repository of source code of the FLOS GIS software.

An improvement made by a programmer can be a part of a product if only s/he meets two requirements. The first one is to publish the source code of improvement done by the programmer. It is not possible to progress next step without meeting this requirement. The second requirement is that the new improvement must correspond with the interest of the community. New improvement must be functional for the community and it should not affect performance of the software negatively. New improvements done by the programmer can be integrated into the main code of the software only by community developers. This is exactly fundamental difference between programmer and community developer. Improvement made by programmer get lost after a time as long as it is not integrated into software's code or published as a FLOSS project. The most permanent way for code contribution to FLOSS world is to share the improvement with FLOSS project's community or to transform the improvement as a FLOSS project and create a community around this project. Today, FLOSS world provides all tools required to transform any development into a FLOSS project to all programmers free of charge. When FLOSS project is setup, it becomes possible to work with volunteers all around the world at any time and to form an open and expandable team with external developers.

More collaborative development environment should be setup in the project in order to continue to develop the FLOSS product. Many developers from different countries can work together in the same source code repositories through Internet connection. This can be possible with the version control system. Concurrent versioning system, Git and Subversion are several FLOSS product examples that are mostly used for setting up version control system in the FLOSS projects. Subversion has become more popular FLOSS product than concurrent versioning system due to being more user friendly application. Version control system enables developers to work as a team on the same source files by keeping track of any changes in the code, integrating the changes that are done by the developers within the existing code, storing all codes with changes as a repository and informing any changes in the code to other developers by sending e-mails. Informing the changes in the code via e-mails as a mailing list allows for a type of passive peer review of changes for other developers (Behlendorf 1999).

To prepare a package/installer is another specialized development task that should also be provided by community members in the FLOSS project. It is one of the time consuming and important service provision for the project. This setup file makes easy for users to install the program to the computer. In order to prepare such a setup file, the source code of the software should be compiled separately for each operating system. The same source code can be used for the compilation in different operating system if the software is developed in the structure of supporting cross platform. After compilation of source code, all produced binary codes in the files are structured with executable file known as setup file, which installs the program to the computer. As a result of this process, the setup file has specific format and structure for the operating system in which the source code is compiled. For example, the extension of setup file is “EXE” for Microsoft Windows OS, “RPM” for Red Hat, Fedora, openSUSE, Mandriva distribution, “DEB” for Debian and

Ubuntu distribution, “DMG” for Apple Mac OS X, etc. If the setup file is missing for your OS or not provided for the users by the project, the user should compile the source code himself or request to do from the developers.

Compiling the source code of FLOS GIS software can be sometimes very tedious and nearly impossible task for the newcomers. While compiling any GIS software, the error can occur in many different ways such as one of the dependencies of the library can be missing, true version of the external library is not used, and a small error can exist in the unstable repository. Each dependent library of FLOS GIS software is developed by different communities and each community tries to enrich the library with many features. Some of the functions in the library can be changed in time with the new version of library. Previous versions of library may become out-dated and not supported by the community anymore. For this reason, new release of all dependencies should be followed, tested and integrated with the application properly. If any error occurs while compiling from the source code of the software, the compiler will immediately stop doing his job and state the error message. The most accurate solution to fix the error is either to ask help from the community via e-mail or to find the most updated compilation instructions that can be found somewhere among the web pages of the project. Although all steps are followed in the instruction, it can be faced with another error message. It is advised to search the similar error messages from the Internet, since nobody can answer your e-mail to help to solve the compilation error if the same problem is explained recently as before. Asking help by using community e-mail list from the packager, release manager or code contributor can be the most effective way to complete the compilation of source code. Once the compiling from the source code of the software is succeeded, it would be used to prepare the setup file.

Requesting help via e-mail is also one of the important issues in the FLOSS projects. There is no guarantee that they will answer, since they are not your workers. There are rules for asking help in the e-mail lists which are explained in the web page of the project. It could be vital for you to obey these rules in order to get proper information from the community. The rules can change from project to project, but there are common rules. The e-mail should be polite, not very long and very informative about the error, and state the version of the source code, compiler, operating system and the environment that is used to compile the software. The error should be posted to the relevant e-mail list.

Another development task in the FLOSS project is to release the software often, since the distribution of the software is done through the Internet. This gives opportunity to revise the new release of the software very easily and cheaper way. When several bugs are resolved, the only thing to distribute the software having less error to their users is to put the new installer in the Internet. For this reason, releasing new version is very speedy in FLOSS environment.

The term “release” may refer to different branches in the source code of repositories for the same version of the software. Before publishing the official stable version, there may be several release candidates. Deciding to use which one of the release such as stable, testing and unstable or release candidates is important in terms of stability and choosing the right version of the dependencies. In general, the stable version of the source code is preferred. There may be stable, unstable and experimental versions. Other communities can call different words for the type of releases like experimental, testing, nightly build, etc. For example, Debian (2008) system prefers to use the term “stable”, “testing” and “unstable”. Although wording is sometimes different, their meanings are almost the same. The term “stable” is the latest official and production release of the distribution, which is primarily recommended to use for the users. Debian (2008) defines the term “testing” distribution as that “the distribution contains packages that have not been accepted into a stable release yet, but they are in the queue for that”. The main advantage of using this release is that it has more recent versions of software. The “unstable” release is where active development of the distribution continues. Generally, this release is run by developers and those who like to live to get error messages. It may be more secure and easy to work and develop the stable release of the product.

3.4. FLOS GIS Libraries and Applications

Available Free/Libre and Open Source (FLOS) development resources like libraries, applications and projects are enormous in FLOSS environment. Many of them are free. First meaning of the term “free” is priceless and second one is to have a chance to read, modify and distribute the code. There are hundreds of FLOS GIS libraries and applications in FLOSS environment. Each has different license, capabilities and features. FreeGIS.org, OpenSourceGIS.org, MapTools.org, GISDevelopment.net and Open Source Geospatial Foundation are several examples for GIS web portals that support the FLOS GIS software products. Other resources related with FLOS GIS libraries and applications are researched by the studies carried out by Ramsey in 2007 and Steiniger in 2008, and Cascadoss Project. In order to decide the most suitable FLOS GIS software, the aforementioned resources like GIS web portals, research studies and projects are searched. The aim of the research is to find the most well known and used FLOS desktop GIS software product.

FLOS GIS software products are presented as a list at FreeGIS.org and OpenSourceGIS.org. In these lists, various GIS software can be found ranging from having limited functionality to advance ones. In September 2008, 339 FLOS GIS software and in August 2010, 351 FLOS GIS software were listed at FreeGIS.org. This number was 238 at OpenSourceGIS.org in May 2007. After 16 months, the number increased to 256 in September 2008. These figures and increase rates show how great effort was put into development of FLOS GIS software and the degree of importance given by the developers in the FLOSS world.

MapTools.org is the web GIS portal announcing new developments about FLOS GIS projects and hosting several of these applications. Several important applications listed at MapTools.org are grouped under three headings as follows (MapTools 2008):

- *Web Tools (CartoWeb, Chameleon, Fusion, ka-Map, MapBender, MapBuilder, MapServer, MapStorer, OpenLayers, OWTChart, PHP MapScript)*
- *Desktop Tools (GRASS GIS, OpenEV, QGIS, uDig)*
- *Utilities and Libraries (AVCE00, DGNLib, GDAL, GeoTiff, MITAB, OGR, PROJ.4, Shapelib)*

GISDevelopment.net is another web GIS portal. The portal listed several FLOS GIS software products in alphabetical order without any classification under free GIS software category in 2008 as follows; OSSIM Image Processing, degree, GRASS GIS, gvSIG, ILWIS, JUMP GIS, MapWindow GIS, PostGIS, QGIS, SAGA GIS, Topology Framework .NET and uDig (GISDevelopment 2008). OSSIM Image Processing program is a Lesser GPL FLOSS product developed for Remote Sensing. Degree is a server based web GIS application. PostGIS is a plug-in which spatially enables PostgreSQL database. Topology Framework .Net is a software development library. Other listed products are FLOS desktop GIS software like GRASS GIS, gvSIG, ILWIS, JUMP GIS, MapWindow GIS, QGIS, SAGA GIS and uDig.

Another researched resource is a study conducted by Steiniger in 2008 which compares FLOS GIS software in detail according to various specifications. FLOS desktop GIS software products evaluated in this study was listed in 2008 as follows; GRASS GIS, QGIS, uDig and derivatives; JGrass and DivaGIS, gvSIG, Kosmo, SAGA GIS, ILWIS, MapWindow GIS, JUMP/OpenJUMP and derivatives; JCS, Single JUMP, Pirol JUMP, DeeJUMP, SkyJUMP (Steiniger 2008).

Also another study has been conducted by Ramsey in 2007. FLOSS GIS software products have been classified according to the used programming language as follows (Ramsey 2007):

1. *Projects developed by C*
    1. *Libraries (GDAL, PROJ.4, GEOS, Mapnik, FDO)*
    2. *Applications (MapGuide, MapServer, GRASS, QGIS, OSSIM, GMT, PostGIS)*
2. *Projects developed by Java*
    1. *Libraries (JTS Topology Suite, GeoTools)*
    2. *Applications (GeoServer, deegree, OpenJUMP, gvSIG, OpenMap, uDig)*
3. *Projects developed by .Net*
    1. *Libraries (NTS, Proj.Net, SharpMap)*
    2. *Applications (WorldWind, MapWindow)*
4. *Web GIS (products publishing geographic data over Internet)*
    1. *Libraries (MapBuilder, ka-Map, OpenLayers, MapBender, CartoWeb)*
    2. *Servers (TileCache, FeatureServer)*

As a result, available FLOS desktop GIS software products listed in several GIS Web Portals like GISDevelopment.net and MapTools.org, and research studies carried out by Ramsey in 2007 and Steiniger in 2008 are summed in Table 3.3. GRASS GIS, QGIS and uDig are three software products which exist in all lists. Following software products like MapWindow GIS, gvSIG and JUMP GIS exist almost in all lists; missing only in one of the lists. So, these applications can be accepted the most well known and used FLOS desktop GIS software products.

Table 3. 3. FLOS Desktop GIS Software Products Mentioned in GIS Web Portals and Research Studies

FLOS Desktop GIS Software Products	**Ramsey 2007**	**Steiniger 2008**	**GIS Development**	**Map Tools**
GRASS GIS	x	x	x	x
QGIS	x	x	x	x
uDig	x	x	x	x
MapWindow GIS	x	x	x	
gvSIG	x	x	x	
JUMP GIS	x	x	x	
SAGA GIS		x	x	
ILWIS		x	x	
Kosmo		x		
OpenEV				x
WorldWind	x			
GMT	x			
OpenMap	x			

In order to decide which FLOS GIS software is the most advance developed desktop GIS application, it is needed to find a study that is carried out by comparing the different aspects of the products and evaluating the projects according to more objective criteria. This kind of study is carried out by The Development of Transnational Cascade Program on Open Source GIS and RS Software for Environmental Applications (Cascadoss Project), which is financed by the European Commission under the Sixth Framework Program, Identification of New Methods of Promoting and Encouraging Transnational Technology Transfer.

Cascadoss Project evaluates GIS and Remote Sensing (RS) FLOSS products according to three evaluation criteria; namely, marketing, technical and economical potential. Marketing potential refers to the maturity of the project, the strength of community, level of support, existing market share, the business options that the license makes possible and collaboration with other projects. Technical potential depends on the software quality which is assessed according to the Quality Requirement Definition of ISO 9126 like the generalized user needs and the typical environment for the software provided by GIS and RS FLOSS products. The last criterion, economical potential refers to how economical is the adoption and operation of FLOSS software in subject comparing to proprietary ones (Cascadoss 2007b). The evaluated GIS and RS FLOSS products are grouped under four headings; namely, Desktop Applications, Development Libraries, Server Application and Environmental Applications. The list of evaluated FLOSS products is shown below according to project classification:

1. *Desktop Applications*
    1. *GIS/RS App. (GRASS, gvSIG, OSSIM, SAGA GIS, OpenEV, Fmaps)*
    2. *GIS App. (QGIS, uDig, Thuban, Kosmo, JUMP, OpenMap)*
    3. *RS App. (ILWIS, ISIS, Octave, RAT, IVICS)*
2. *Development Libraries*
    1. *RS Library (GDAL, GSF, IVICS, ORFEO, PROJ.4)*
    2. *GIS/RS library (tclSADIE)*
3. *Server Application*
    1. *Web Services (deegree, GeoServer, MapGuide, MapServer)*
    2. *Web Tools (OpenLayers, MapBuilder, Mapbender, Chameleon, CartoWeb, ka-Map, kvwmap, FlashMapping)*
4. *Environmental Applications*
    1. *GRASS, gvSIG, HidroSIG, SAGA GIS, ILWIS, uDig, Virtual Terrain Project, Calypso Simulation Platform, EpiGrass, r.terraflow, DIVA GIS and Basins*

The main subject of this thesis is FLOS desktop GIS software products. For this reason, the software products which belong to only Remote Sensing field are not included in the thesis. Both categories; GIS/RS and GIS software products under Desktop Applications are reviewed according to three evaluation criteria in which each score of software products are shown in Tables 3.4-3.6.

Table 3. 4. Marketing Potential of FLOS Desktop GIS/RS and GIS Software Products (Cascadoss 2007b)

Type	Software	Maturity of project	Strength of Community	Market Share	Licence issues	Collaboration with projects	Total
	Max. Score	*15.0*	*15.0*	*12.0*	*9.0*	*9.0*	*60*
GIS/RS	**GRASS**	15.0	14.3	12.0	9.0	9.0	**59.3**
	OSSIM	11.7	9.0	12.0	9.0	9.0	**50.7**
	OpenEV	10.7	9.4	12.0	9.0	9.0	**50.1**
	gvSIG	12.8	9.0	10.0	9.0	9.0	**49.8**
	Saga GIS	11.3	5.6	10.0	9.0	9.0	**44.9**
	Fmaps	3.6	0.8	4.0	9.0	0.0	**17.4**
GIS	**QGIS**	14.0	11.6	12.0	9.0	9.0	**55.6**
	Thuban	14.0	13.1	10.0	9.0	9.0	**55.1**
	OpenMap	14.0	11.1	7.9	6.0	9.0	**48.0**
	uDig	12.5	5.8	6.0	9.0	9.0	**42.3**
	JUMP	10.5	6.8	0.0	9.0	9.0	**35.3**
	Kosmo	6.8	3.0	2.0	9.0	9.0	**29.8**

Table 3. 5. Technical Potential of FLOS Desktop GIS/RS and GIS Software Products (Cascadoss 2007b)

Software	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Total
Max. score	*15*	*9*	*9*	*9*	*9*	*9*	*60*
GRASS	11.7	4.2	8.2	6.9	7.4	8.4	**46.9**
gvSIG	8.8	4.2	5.6	7.9	6.4	6.9	**39.8**
OSSIM	8.6	3.6	5.0	6.4	6.7	6.4	**36.7**
SAGA GIS	5.6	1.8	5.4	6.9	5.6	5.1	**30.4**
OpenEV	4.7	1.2	4.3	5.8	4.5	6.6	**27.1**
Fmaps	1.9	3.0	1.5	6.4	1.2	1.9	**15.8**
QGIS	8.6	3.3	7.4	7.5	7.2	6.7	**40.8**
uDig	5.4	1.5	7.1	5.6	4.6	6.4	**30.6**
Thuban	6.4	2.4	4.7	6.8	4.0	6.0	**30.2**
Kosmo	8.1	0.6	4.4	5.4	6.0	4.1	**28.6**
JUMP	4.2	1.4	3.7	5.4	4.9	6.0	**25.6**
OpenMap	5.7	3.2	3.5	4.5	4.2	4.2	**25.4**

Table 3. 6. Economical Potential of FLOS Desktop GIS/RS and GIS Software Products (Cascadoss 2007b)

Type	Software	Cost of installation	Cost of migration	Cost of operation	Total	Evaluated Version of Products
	Max. score	*24.0*	*18.0*	*18.0*	*60.0*	
GIS/RS	**OpenEV**	24.0	5.9	10.8	**40.7**	1.8
	GRASS	24.0	7.1	9.0	**40.1**	6.2.3
	OSSIM	24.0	5.9	7.2	**37.1**	1.7.4
	gvSIG	24.0	5.9	7.2	**37.1**	1.1.2
	SAGA GIS	24.0	4.8	7.2	**36.0**	2.0.2
	Fmaps	18.0	4.8	5.4	**28.2**	0.0.2
GIS	**QGIS**	24.0	14.0	12.6	**50.6**	0.9.1
	JUMP	24.0	15.0	10.8	**49.8**	1.2
	Thuban	24.0	13.0	10.8	**47.8**	1.2.1
	OpenMap	24.0	13.0	10.8	**47.8**	4.6.4
	Kosmo	24.0	9.1	10.8	**43.9**	1.2
	uDig	24.0	13.0	5.4	**42.4**	1.1-RC 14

As it can be seen from the tables, first, second and third order of having total highest score in the evaluation criteria are GRASS GIS, QGIS and Thuban for marketing potential, GRASS GIS, QGIS and gvSIG for technical potential, and QGIS, JUMP and Thuban - OpenMap (last two have same score) for economical potential, respectively. GRASS GIS and QGIS are two FLOS desktop GIS software products that have almost the highest score in all evaluation criteria.

Finally, Open Source Geospatial Foundation is another important research area for this thesis. This foundation was established with the aim of producing FLOS spatial software, and providing collective development of FLOS GIS projects supported by communities under one umbrella. The Foundation provides an environment where users and developers can contribute development of projects and share their opinions via web portal. The establishment aim of the Foundation is the same with the philosophy of free software act and the foundation challenges to continue this philosophy in GIS field. FLOSS projects supported by the Foundation in 2008 are shown below in a classified manner (OSGEO 2008a):

- *Web Mapping (deegree, Mapbender, MapBuilder, MapGuide Open Source, MapServer, OpenLayers)*

- *Desktop Applications (GRASS GIS, OSSIM, QGIS, gvSIG)*
- *Spatial Libraries (FDO, GDAL, GEOS, GeoTools)*
- *Metadata Catalogue (GeoNetwork)*
- *Other Projects (Public Geospatial Data, Education and Curriculum)*

Within six months time frame, several FLOSS projects are added under the foundation’s umbrella like deegree and MapGuide Open Source in Web Mapping category, gvSIG in Desktop Application and GEOS in Spatial Libraries. These are the sign of how the mapping community is active and has the accelerating development speed. OSGEO (2008b) explains the aim of foundation is that “the foundation provides a legal and administrative framework to support ongoing development of FLOSS projects”.

In January, 2008, the foundation supports three FLOSS projects, namely, GRASS GIS, QGIS and OSSIM under Desktop Application category. The gvSIG project is added later that date under the foundation’s umbrella. OSSIM is an advanced geospatial image processing for remote sensing, photogrammetry and GIS (OSSIM 2008).

As a result, GRASS GIS and QGIS are two products that are mostly listed and researched as FLOS desktop GIS software by various references. The gvSIG, JUMP/OpenJUMP GIS family, Thuban, uDig, MapWindow GIS and OpenMap are also valuable FLOS desktop GIS software products. These products can be downloaded from their web site, installed to the computers in unlimited numbers and are ready to use by the users freely in any time.

CHAPTER 4

DEVELOPMENT OF SDA SYSTEM

4.1. Definition of SDA System

The following opportunities are possible in nowadays such as having sufficient computational power for performing SDA techniques, the ease of use of development tools, the availability of enormous geography related FLOS libraries, having no restriction to use these FLOS resources and the possibility of gaining the support of SDA developers coming from different application areas. These opportunities make possible that the fourth type of coupling problem, the complete integration of SDA techniques in GIS, can be successfully implemented without the support of any vendor in the FLOSS environment. In other words, the complete integration of SDA techniques in GIS can be applied without the support of commercial GIS vendor when FLOSS development methodology is properly applied.

In this thesis, SDA system refers to the development of SDA techniques in GIS in a FLOSS environment. SDA should be integrated with GIS in order to disseminate these techniques not only for researchers but also for GIS users. This thesis succeeded to integrate SDA techniques within the FLOS GIS software as FLOS SDA system in order to provide more functional GIS software for both researchers and GIS users, to get a chance to use these techniques in a user friendly GIS software for users, to learn how the techniques work and to develop new ones for theorists and developers. SDA can be disseminated and facilitated more widely, if their techniques are part of GIS software according to this thesis's coupling strategy.

In general, a generic SDA system should have the following properties:

- performing with graphical user friendly interface in the GIS environment,
- providing a modular environment,

- integrating different SDA algorithms or libraries written with several languages like FORTRAN, C/C++, Java, etc.,
- providing a chance to create a test environment in order to develop new SDA techniques for SDA theorists and developers,
- easy to use,
- cross platform support meaning that the same source code should run in different operating systems like Windows, Linux and Mac OS,
- legally licensed and have no license fee,
- having not only SDA tools but also GIS tools,
- detailed explanatory descriptions supported with examples,
- support of multi language in the GUI,
- a right to access the source code of the SDA techniques, study and modify the program,
- freedom to run the program for any purpose including commercial purposes,
- freedom to copy the program without any cost as much as s/he wants,
- a right to improve the program and share his/her improvements by releasing source code to the public,
- easy installation for different operating systems,
- a right to download the installer from the Internet,
- a freedom to install the program to the computers in unlimited number,
- possibility to develop the program with the contribution of researchers coming from different disciplines,
- a right to access the main repository of the source code of the software in order to add new SDA tools by external developers,
- setting up collaborative development environment for the developers working at the same source code accessible from Internet independent of time and place,
- And last, but not least, interoperable with existing GIS data formats and open standards.

These requirements strive to use FLOSS development methodology. The most obvious advantage of using FLOSS products while developing FLOS SDA system is to discover the SDA techniques by exploring from their source code, possible to integrate with other libraries and no license fee required using all software products. There is a gap in the literature of SDA field about explanation of how these techniques are applied. The explanations for the geocomputation of these analyses are either missing or poorly documented. For the proprietary SDA products, the situation is worse than free software. Different results can be taken for the same analysis that is done in different software with the same data. Finding the reasons why the results are different in the proprietary software is something like digging a hole with a needle. For free software products, if the user has the knowledge of reading source code, s/he can learn the reasons of different results from the source code of the tool.

The development of FLOS SDA system carried out in this thesis has met all requirements of defined generic SDA system except three requirements which are detailed explanatory descriptions, support of multi language in the GUI and setting up collaborative development environment for external developers working in the same code at the same time due to time limitation of the thesis. However, the development study has completed the integration of a set of 14 SDA techniques in FLOS GIS software in the FLOSS environment in 2.5 years. The complete integration of SDA techniques in GIS can be done in the FLOSS environment without the support of GIS vendor, which is the new interpretation of the fourth type of coupling problem. The support of any vendor is not an essential factor in order to achieve the complete integration of SDA techniques in GIS. Next section reviews FLOSS development methodology based on the development of FLOS SDA system.

4.2. FLOS SDA System Development Methodology

SDA techniques are incorporated with GIS in the FLOSS environment in order to disseminate and facilitate the SDA field. The methodology followed during the development of SDA system is summarized as shown in Figure 4.1. At first, the reasons of why SDA could not been coupled with GIS are reviewed and then, the requirements of SDA software are defined. These requirements can be met in the FLOSS environment. The need for FLOSS development is explained. The meaning of free software and open source are studied. After then, the FLOS GIS Software, spatial libraries, applications and tools are tried in the Linux operating system. In order to achieve this integration in the FLOSS environment, at first, the development has started in Linux operating system and then, continued in Microsoft Windows OS. By this way, cross platform support of the system is sustained. Which FLOS GIS software products should be used as the main GIS component of the system is selected and then, other components of the system are decided. When the structure of system is completed with selected components, the inner architectural design schema of the system is evolved in time and revised several times in order to reach more mature and stable version. When the source code of the development is distributed with developed repository through Internet and opened to use with QGIS community, FLOSS development methodology is started to be applied. The information about the development is shared with the users in the prepared web site. This methodology has established for this thesis to provide a FLOS SDA system which couples SDA techniques within user friendly GIS software product by using FLOSS products. By this way, the usage of the developed system has reached to more than 1600 users.

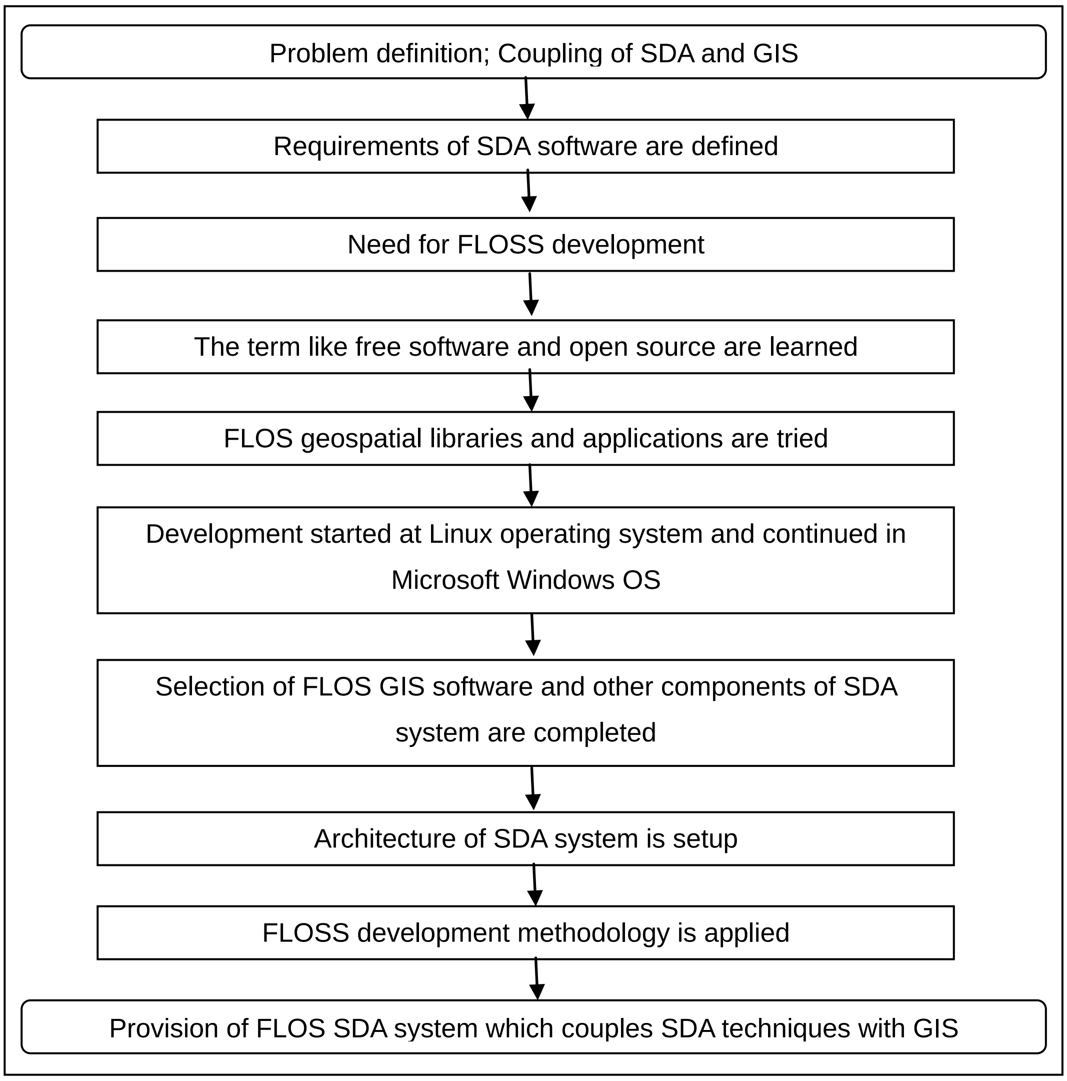

Problem definition; Coupling of SDA and GIS
Requirements of SDA software are defined
Need for FLOSS development
The term like free software and open source are learned
FLOS geospatial libraries and applications are tried
Development started at Linux operating system and continued in Microsoft Windows OS
Selection of FLOS GIS software and other components of SDA system are completed
Architecture of SDA system is setup
FLOSS development methodology is applied
Provision of FLOS SDA system which couples SDA techniques with GIS

Figure 4. 1. Summary of FLOS SDA System Development Methodology

The work carried out during the development of this system in this thesis has started with the learning of Linux operating system. It may be difficult to get accustomed to learn and use the Linux at the beginning. Newcomers face with new interfaces and manner of usage. Although there are windows based graphical user interface for some of the tasks related with managing the operating system, it still requires to use command prompt in parallel with graphical user interface. It requires a time to learn Linux operating system for the person who gets custom to work only in Windows operating system. Learning period changes from person to person. In this thesis, it is required to spend approximately 6 months. In this time frame, several Linux distributions like Mandriva, Fedora, Debian and Ubuntu are personally tried in the first half year of 2007. Ubuntu distribution is found as the most user friendly Linux distribution. First development stage has started in Ubuntu distribution. After then, development environment has shifted to setup in the Windows operating system. With the learning from Linux, the main development environment has become in Windows. Linux knowledge has helped the developed FLOS SDA system to have cross-platform support. Ubuntu distribution has become a test environment for the developed system to ensure that the system is also working in the Linux. Linux knowledge is also served to solve the programming differences between two operating systems faced during the development of the system at specific points like defining folder path, GUI clearing, refreshing plotting mechanism and output of command prompt for logging purposes.

Next step is to decide which FLOS GIS software products should be used. The most widely known FLOS desktop GIS software products are searched from the research studies carried out by Ramsey in 2007 and Steinger in 2008, and GIS web portals like GISDevelopment.net, MapTools.org, FreeGIS.org and OpenSourceGIS.org, and Open Source Geospatial Foundation and Cascadoss Project. According to findings from the research of these resources completed in 2008, QGIS is selected as the primary desktop GIS component of the developed FLOS SDA system. The architectural schema of SDA system is setup in 2008 and the inner design of schema has changed several times in 2.5 years in order to create a systematic of adding a new tool and making the development more stable as changes in the QGIS interface.

The next step is to decide SDA related components of the system. Tremendous development effort is ongoing on the development of spatial libraries and applications. There are a lot of scripts, spatial libraries and applications. Feature Data Object Data Access Technology, GDAL, GEOS, GeoTools, MetaCRS, PROJ.4 and PostGIS are some of the examples for comprehensive and advanced spatial libraries. GDAL and PROJ.4 are two FLOS spatial libraries used in the developed FLOS SDA system. GDAL supports many raster and vector GIS data formats. PROJ.4 deals with the geographic and projected coordinate systems. It is tried to find SDA library among these kinds of libraries. However, the main motivation of these libraries are rendering of spatial data, GIS data formats, coordinate systems, database back-end, GIS utilities and spatial data infrastructure, but not the SDA field.

Mostly, SDA techniques are scattered all over in the statistical packages and numerical computing environment such as S-Plus, Matlab, etc. The SDA techniques are attached to these environments as add-on, module, toolbar or extension. For example; S-Plus has SpatialStats module and Tests for Spatial Correlation package. However, all these examples are closed source applications. This means that the SDA techniques could not be studied from their source code. To access the source code is the key factor for testing the techniques and developing the new ones. This kind of barrier breaks down one of the objectives of the SDA system.
When SDA system is developed with proprietary statistical package and GIS software, the use of the system could be limited due to license fees. To pay both proprietary statistical package and GIS software license fees would be costly for the users. For example, the license fee of ESRI ArcInfo licensed ArcGIS Desktop is $30,000 plus value added tax for commercial usage in Turkey in 2010. License fee is charged for each copy of application in the closed source environment which breaks down other objectives of SDA system which are that the system should be free to use and free to redistribute among the users. For these reasons, proprietary statistical packages could not be used as a SDA component in the development of SDA system. Instead of proprietary statistical package, R, FLOS version of S-Plus, is used as the SDA component of developed system.

R is an environment and a programming language. The processes in the R environment are done with the command line interface which is not user-friendly and requires sort of the knowledge of programming. R is not appropriate for the GIS users who have no programming knowledge. R is not a GIS software product and has got a different mentality and style of handling spatial data that is too much complicated and different for many GIS users, who have get accustomed to use user friendly menu driven GIS software.

The development of FLOS SDA system is made up with the following components; the development framework is Qt, FLOS GIS Software is QGIS, FLOS statistical package is R and additionally used two FLOS spatial libraries are GDAL and PROJ.4. The last important component is the programming language itself which is Python. Python programming language is high-level object oriented and dynamic programming language. The most important advantage of using Python for the development of SDA system is to call any function that is available in another library or application which are developed with another programming language. This ability of Python is known as the extendibility property of Python programming language and can be called as wrapping or binding. Wrapping is required in order to communicate two different programs with each other; namely QGIS and R. SWIG, Boost Python and SIP are the software development tools that connects applications written in C and C++, and communicates with Python. Wrapped C or C++ modules can be called with Python code. The development framework of QGIS is Qt. The Qt framework is wrapped with SIP tool. In order to create seamless integration, QGIS is also wrapped with this tool. In SIP terminology, wrapped modules are often called bindings for the library. SIP creates Python bindings for QGIS library. A binding of Qt framework is named as PyQt and QGIS is PyQGIS. Another advantage of using Python programming language is to embed the interpreter of Python in any application. Python is embedded in QGIS. By this way, Python codes can be interpreted in the QGIS. With the extendibility and embeddable properties of Python, QGIS and R application's classes and methods can be called with Python codes.

Another advantage of using Python is to have world-wide support. With this support, Python is used in many application domains. There were 12,616 Python modules in December 2010. RPy2, one of Python module, is also used in the development of SDA system. Similarly, RPy2 is used to bind the R language and its external packages. Another Python module required only for Microsoft Windows OS is PyWin32 which is the Python extensions for Windows OS. NumPy is also optional Python module which speeds up the array processing for numbers, strings, records and objects.

Python is used as “glue” language in the system. Python sticks to QGIS with binding of PyQGIS and to Qt4 with binding of PyQt4 with the help of SIP tool, and to GDAL and PROJ.4 through R with RPy2 interface. Each component of SDA system; QGIS, PyQGIS Bindings, Qt, PyQt Bindings, Python, Python modules, R and R packages are explained in detail in “Architecture of SDA System and Properties of Components” section of this chapter.

After the architectural schema of SDA system is determined, the FLOSS development methodology is followed. SDA system source codes are published through Internet and can be accessible in any time during the development. The visible part of SDA system is the development of plug-in in the QGIS. Plug-in is the outcome development of the system, because this thesis's coupling strategy is to integrate SDA techniques in the GIS software. The plug-in is named as SDA4PP which is the abbreviation of “Spatial Data Analysis for Point Pattern”. The implemented techniques in the system are based on point pattern analysis. This limitation is done, because SDA field is so much wide and covers many different techniques. In order to overcome this variety, the implementation is limited according to the subject of SDA. Adding new techniques is systematically designed in the inner architectural schema of the plug-in. This schema can be applied not only techniques of point pattern analysis but also other subjects of SDA field. To add one more SDA technique is only related with spending more time and effort.

SDA4PP plug-in is revised 196 times in 2.5 years. The source code of plug-in can be accessed through repository and shared with QGIS community. The plug-in is mainly tested with this community. Some of the revisions are done to fix the bugs reported by the community, some of them are revised to add new tools and some are to change the inner structure of the design.

A fetching mechanism is established inside of the QGIS that is to communicate between the user and repositories. By this way, installing and updating the plug-in has become more easy and user friendly way for the users. Repository is formed in the structure of Extendable Markup Language (XML). One XML file structures the properties of plug-in. Communication between QGIS and repositories are done according to this file. XML file paths to the compressed zip file which has Python plug-in source code. The linkage mechanism downloads the zip file through Internet and extracts the zip file into the specific path of user computer. The plug-in code is interpreted by embedded Python interpreter to make ready to use the plug-in in the QGIS. If an error is found while interpreting the code, the plug-in is disabled. Interpretation also covers to check whether all dependencies of plug-in exist and can be imported or not. If the interpretation is successful, the plug-in is ready to use in the QGIS.

FLOSS development method requires publishing the source code of the development, giving information about how the development is ongoing, and attracting the users and developers to help to the development process by adapting each person as a member of the project. Source code is published under a dedicated repository which is prepared to realize this specific purpose. Information about the development is provided mainly in prepared web site, which is devoted for the development of the system. Explained facilities such as revision history, development environment, snapshots, blog, reporting a bug, etc. are setup in the web site to inform the user about the development.

Home page of web site has links to blog, wiki, and stores several pages like report a bug, repository and download pages. Announcements about the new release of system, plug-in and installer are announced in the blog. Report a bug web page keeps the information related with the error that is reported by the users. Some of the bugs are also reported via e-mails. Bugs are recorded by using the infrastructure of code.google.com. Repository web page stores the source code of the plug-in and is created with one XML file formatted with extensible stylesheet programming language. The plug-in source code can be downloadable from the repository page. Download web page has the installer which is prepared to ease the installation of the SDA system to Microsoft Windows OS. Installer is developed by using Nullsoft Scriptable Install System. Installer gathers the installer of each component of the system into one executable setup file. While installing the components of the system, the installer adds the path of R statistical package into path variable in the Windows OS. Installer is prepared for only Windows OS. For Ubuntu distribution, the installation instruction is given in the page. The instruction explains how to install the SDA system with command line interface in Ubuntu distribution.

Other pages in the web site are revision history, development environment and snapshots. Revision history web page lists the reasons of revision, version number and revision date. Development environment page displays all dependencies of the system with their versions. Snapshots web page shows the GUI and analysis results of integrated tools. Snapshots web page also demonstrates how the integrated tools are evolved through revisions and the number of tools are increased in time. All web pages in the site help to explain how the system has developed in time.

The most important meaning of FLOS is to create a collaborative environment for developers as well as users. There is no direct message in the web site of SDA system for both user and developer that development of the system requires any help or demanding collaborative development with other programmers. This choice is done deliberately in order to preserve the originality of the work. The development work is carried out within the scope of thesis. In addition to this, no request has come from any person to develop the system together. Absence of FLOS collaborative tools in the web site may cause to support this situation stronger. Within the scope of this extent, FLOSS development methodology is applied in this manner for the development of FLOS SDA system during 2.5 years from the learning of QGIS community.

4.3. Linux Distributions

Linux distributions are searched to decide which one of them should be used in order to support cross platform while developing FLOS SDA system. Selected distribution is used to create a test environment for developed SDA system. Whether the developed SDA system is working or not is tested in this selected distribution. By this way, it is guaranteed that the system is working in both operating systems; Windows and Linux.

There are hundreds of Linux distributions. The most widely used and user-friendly distribution is searched. The exact number of Linux distribution was 348 in June 2008 at DistroWatch.com (Distrowatch 2010). Linux distributions can be used in different forms; from desktop and server operating system to embedded system as well as booting from a floppy disk. Well known distributions can be listed as Red Hat, Fedora, Debian, Ubuntu, Mandriva, SUSE, Gentoo and Slackware. Red Hat, now, is in the business market and one of the oldest. It has also free distribution, which is Fedora. Debian is the most popular, one of the oldest and mature distribution. SUSE distribution is the Novell Company’s answer to Red Hat and in the business market like Red Hat. Free version of SUSE is openSUSE. Slackware based distributions are mostly for advanced Linux users and servers. Ubuntu is relative late comer Linux distribution but has gained fame for focusing on immediate usability on the desktop and tag lines as “Linux for human beings”. The power of Ubuntu comes from the Debian. Ubuntu is Debian-derived distribution and is the best place for complete new comers to start. Top ten Linux distributions are listed in Table 4.1 according to average number of hits per day. This statistic is calculated by Distrowatch according to the counting of all visits on the main site, as well as on mirrors. Only one hit per IP address per day is counted and figures are updated daily (Distrowatch 2010). The most widely visited Linux distribution in May 2008 is Ubuntu for three time frames; last 1, 6 and 12 months as seen in Table 4.1.

Table 4. 1. Most Widely Visited Linux Distribution for Last 1, 6 and 12 Months Periods in May 2008 (Distrowatch 2010)

Last 1 month			**Last 6 month**			**Last 12 months**		
Rank	**Distribution**	**Visits**	**Rank**	**Distribution**	**Visits**	**Rank**	**Distribution**	**Visits**
1	Ubuntu	2,056	1	Ubuntu	2,348	1	Ubuntu	2,324
2	openSUSE	1,863	2	openSUSE	1,485	2	PCLinuxOS	2,115
3	Fedora	1,842	3	Fedora	1,382	3	openSUSE	1,510
4	PCLinuxOS	1,175	4	PCLinuxOS	1,360	4	Fedora	1,321
5	Mint	1,100	5	Mint	1,314	5	Mint	1,216
6	Mandriva	835	6	Mandriva	935	6	Sabayon	951
7	Debian	784	7	Debian	834	7	Mandriva	863
8	Puppy	764	8	Sabayon	730	8	Debian	829
9	OpenSolaris	693	9	Dreamlinux	683	9	Damn Small	679
10	Sabayon	673	10	Damn Small	671	10	MEPIS	646

Fedora, Debian, Mandriva and Ubuntu are personally tried. Ubuntu has been found the most user friendly Linux distribution on the desktop. The community support is relatively fast. There are great numbers of FLOSS packages in the specific format “DEB” in Ubuntu repositories due to being the Debian-derived distribution. Ubuntu is chosen as Linux distribution to be used to test whether the developed SDA system will work or not also in Linux operating system. By this way, the cross platform is supported with the developed SDA system in this thesis. Which Linux distribution is used for this development is not so much important. In order to achieve this development, any of them can be chosen, because each of them has the same Linux kernel. Each distribution is designed to run the software properly when the source code is successfully complied in that distribution. However, building from source code can be difficult in Linux environment, because most of the GIS software have many dependencies. Each dependency should be either complied or linked with main source code of software to work together. In addition to this difficulty, it may be required to use exact version of dependent library which is compatible only with specific version of GIS software. So, while building the source code of GIS software, it is not a good idea to use the latest version of library in order to have the new features of libraries. Which versions are compatible with each other is mostly known by the community developers of GIS software.

Building from source code is mostly documented. However, the documentation may become out-dated in terms of the specified version of dependent libraries and the version of GIS software. While building the GIS software from source code, to know exact version of each dependency could be essential. When the compiler gives an error, it may be related with using wrong version of library. So, a help should be asked by sending a polite and informative e-mail to the community. E-mail should have the version of operating system, complier, GIS software, the dependencies and the error message. The error may be a known and reported as a bug to the GIS community. But, GIS community could not solve the problem, because this bug could only be fixed by other community developer, and that developer could have other priorities. At the final stage, all bugs are fixed, but it requires time.

FLOS GIS software has many dependencies. For example, QGIS software's dependencies are Flex, Bison, GRASS, GEOS, GDAL, SQLITE, GSL, PROJ.4, PostGIS and Expat. Each dependency should be installed in the Linux distribution. Actually, each dependency is developing in another FLOSS project with another community. They have separate installer, compilation process and other dependencies. Compiling each dependency from source code requires a lot of time. For this reason, it can be vital for the developer whether used Linux distribution have an installer for each dependency of FLOS GIS software or not. Instead of dealing with the difficulties of compilation, these installers can be used while installing the FLOS GIS software. Choosing user friendly Linux distribution and having an installer for variety of application in distribution's package repositories are very important for developer which makes the development work easier. To have an installer for each dependency of FLOS GIS software is an important criterion while deciding to select which Linux distribution should be used. The installer for each dependency of the components of SDA system is provided for Ubuntu distribution. This distribution is appropriate for developing SDA system in terms of providing user friendly environment and supporting huge variety of application.

4.4. Reasons to Choose QGIS as Main GIS Component

Several GIS web portals and researches are examined in order to find the well known, mostly used and advance FLOS desktop GIS software. QGIS and GRASS GIS are two FLOS applications that are mentioned in all searched resources such as MapTools.org, GISDevelopment.net, Cascadoss Project, Open Source Geospatial Foundation and research studies conducted by Steiniger in 2008 and Ramsey in 2007. QGIS is selected as the main GIS component of the SDA system according to research done in the mentioned resources.

The most important selective decision is taken according to the foundation whether they are supporting the project or not. The foundation was supporting only GRASS GIS, QGIS and OSSIM as the FLOS desktop GIS software products in January 2008. In September 2008, gvSIG was added in the desktop application category as the new incubation projects by the foundation. OSSIM software is a FLOS remote sensing product. The gvSIG project supported by the foundation after the implementation started in the thesis. Therefore, two software products; namely OSSIM and gvSIG, were excluded from the selection of the main GIS component of FLOS SDA system.

The gvSIG project started at the end of 2003 to be developed with the financial support of Spanish government. The regional administration has planned to change all computer infrastructures to Linux environment. The purpose of developing gvSIG is to create a FLOS desktop GIS software product like having features of ESRI ArcView GIS series 3 free of charge for the FLOSS world. The project is trying to create GIS software as powerful as aforementioned proprietary GIS software for municipalities that will use the software in the Linux operating system. This aim has been accomplished in spite of several drawbacks. Though the project is supervised by only one country, participation to community is at international level. Disadvantage of this software is lack of sufficient documentation for developers in 2008. Moreover the software is dependent on about hundred C++ and Java libraries. This makes difficult to develop gvSIG software by the developer who are not member of the community.

QGIS and GRASS GIS have taken highest scores in the evaluation of Cascadoss Project. At the last comparison, QGIS has been chosen instead of GRASS GIS. When the history of GRASS GIS is reviewed, the reason can be easily understood why GRASS GIS has not been chosen in spite of its power and widespread use. GRASS GIS was started to be developed by US Army Construction Engineering Research Laboratories on UNIX environment in 1982. The aim for developing the GRASS GIS between 1982 and 1995 is to create an alternative against proprietary desktop GIS software of that time and lower the prices. Development of GRASS GIS was terminated due to the prices decreased by the release of ESRI ArcView GIS series 3. After then, GRASS GIS Software has started to be developed in 1997 as the first FLOS GIS project. When GRASS GIS and QGIS are compared in terms of their capabilities, GRASS GIS has more comprehensive content. However, all tools of GRASS GIS can be called via community developed plug-in and executed within the user friendly graphical interface of QGIS.

The architecture of GRASS GIS is old and complicated since it has long software development process. Appearances of GUI and window system are based on old-fashioned UNIX. It takes a long time for users familiar with Windows operating system to adapt them to this system composed of many separate window sections and command lines in UNIX style. GRASS GIS community attempts to develop a new version to overcome this out of date architecture by integrating wxPython GUI library. The wxPython is cross platform GUI toolkit developed with Python programming language that wraps wxWidgets GUI library written in C++ (wxPython 2010). With this library, GRASS GIS looks like native view in which operating system is installed. Another aim of developing this version of GRASS GIS is to produce native binary code for Microsoft Windows operating system. Until now, GRASS GIS can be run in Windows OS with the help of emulator, Cygwin, which creates temporarily Linux environment for the software to run in the Windows OS. Since these types of renewals can affect the source code of GRASS GIS, developing SDA techniques in GRASS GIS could not be stable until these improvement studies have finalized.

In order to decide which software is the most mature, well known and used desktop GIS software, it is needed more objective statistics about GRASS GIS and QGIS. The available statistics about FLOS software could be found by Ohloh Project. The aim of Ohloh Project is to connect people through the software they create and use. Ohloh Project is online community platform that aims to map the landscape of FLOS software development. In September 2008, the site lists 12,820 FLOS projects. Ohloh Project retrieves the statistical data from revision control repositories. This statistic gives information about the longevity of projects, their licenses and lines of source code. The comparison is done according to "project cost" category of the project. Ohloh Project defines this category as used for "estimating how much it would cost to hire a team to write the software from scratch". According to the Ohloh Project's results, the comparison of two FLOS GIS projects like QGIS and GRASS GIS is summarized in Table 4.2. Required efforts to develop GRASS GIS is almost 4 times bigger than QGIS in terms of the number of line in source code, the number of person in a year and total cost in September 2008. This situation is the result of early commencing the software development of GRASS. This project started in 1982, while QGIS is in 2002. This causes that the number of available tools in GRASS has outnumbered the QGIS. However, this disadvantage has overcome from QGIS community by embedding those tools in QGIS. Those tools can also run from inside of QGIS. Moreover, these figures are changed in favor of QGIS in August 2010. The number of codebase, estimated personnel effort and total cost of QGIS project have tripled the GRASS GIS development efforts. QGIS has increased more than 10 times estimated personnel effort, written code line and total cost between September 2008 and August 2010. However, the same figures for GRASS GIS have decreased slightly within the same time period.

Table 4. 2. Ohloh Project Figures for QGIS and GRASS GIS in 2008 and 2010 (Ohloh 2010a and 2010b)

Project Cost Category of Ohloh Project	**QGIS**		**GRASS GIS**	
	2008	**2010**	**2008**	**2010**
Number of Code Line	161,122	1,585,531	576,369	551,458
Estimated Effort of Person in a year	42	447	156	149
Total Cost ($)	2,283,668	24,566,207	8,599,648	8,204,984

Ohloh Project (2010a) states that "QGIS is mostly written in C++, has mature and well established codebase, increasing year over year development activity, and has large and active development team". Ohloh Project (2010b) states that "GRASS GIS is also mostly written in C, has mature and well established codebase, and has large and active development team, but decreasing year over year development activity".

The major improvement done in the structure of QGIS in 2008 is to add Python programming language with version 0.9.0. This improvement has made to outnumber the figures and grow the development rapidly. The trace of growing development efforts can also be seen in the number of QGIS core plug-in. The number of QGIS core plug-in version 0.9.0 is 11 and version 1.5.0 is 23. The number of developed core plug-in has doubled in two years.

Early commencing the software development of GRASS GIS leads to its architecture to compliant with POSIX which is variants of UNIX operating system like Linux, FreeBSD, etc. However, this is disadvantageous for the Microsoft Windows OS. In order to operate GRASS in Windows OS, there should be an emulator like Cygwin. This would be difficult for users who are custom to have user friendly GIS software in Windows OS. Many POSIX's oriented products differs from windows based applications in terms of programming and mentality; ease of usage and terminology. That could be a problem for GIS users who get custom to use user friendly menu driven GIS software in the Windows OS. Another disadvantage is the appearance of GUI of the application which does not have a native look like of installed operating system.

Since the development of QGIS started in 2002, QGIS has advanced architecture and is relatively more up-to-date, which makes easier to develop with QGIS. Since there is no need to make fundamental changes in software architecture as in GRASS GIS for Windows OS, it can be claimed that development environment of QGIS is much more stable with respect to GRASS GIS. With release of QGIS version 1.0, the code stability in the QGIS library has become more mature level. In brief, QGIS was chosen as the main GIS component of SDA system to be developed in this thesis due to its features listed below:

- Supported by Open Source Geospatial Foundation
- Developed by a broadly-participated international community
- Having GNU GPL
- Developing in a modular structure
- Accessible descriptions for the methods and classes in QGIS library through the Internet

- Development possibility for creating a new FLOS GIS software or a plug-in
- Using all tools of GRASS GIS within QGIS user friendly interface
- Availability of detailed and updated manuals
- Having extensive documentation
- Supporting Python programming language which provides Rapid and Agile Application Development
- Having a newer software architecture with respect to GRASS GIS
- More user friendly GUI
- Having multi platform support
- An active project and developing quickly

The main reason of selecting QGIS is that QGIS enables to extend its functionality with plug-in architecture by using also with Python programming language. User contributed plug-in in the QGIS can be created by using Python or C++ programming language. Even new GIS software can be created by QGIS library with both languages. But, the aim of this thesis is to couple SDA techniques in the GIS software. For this reason, instead of developing a new GIS software, a new plug-in is developed in QGIS, which is named as SDA4PP that is the abbreviation of Spatial Data Analysis for Point Pattern. The implementation details of this plug-in are explained in the following chapter.

Developing a plug-in with Python programming language fits with the aim of this thesis. Developing with Python is surely much easier than with C++ programming language, because interpreting the plug-in with Python on every change is much faster than compilation of C++ programming language. Python has much shorter code than C++ which makes a less error prone code (Langtangen 2006). Python has faster development environment which matches with the Rapid and Agile Application Development strategies. For these reasons, QGIS is chosen as the main GIS component of the SDA system in which the integration of SDA techniques is implemented. The properties of each component forming the whole SDA system are explained in detailed in the next section of this chapter.

4.5. Architecture of SDA System and Properties of Components

QGIS, Qt and R are the main components of the developed FLOS SDA system. All of them can be called together with several python bindings, PyQGIS, PyQt4 and RPy2, respectively. PyQt4 is used for GUI development of SDA system. RPy2 is used to bind the statistical package. Python programming language is used as the "glue" between these products. For statistical part, four external R packages are used, namely; "sp", a package for spatial data regulating the structure of spatial class and methods under one package, "maptools", tools for reading and handling spatial objects, "spatstat", a package about spatial point pattern analysis, model-fitting, simulation and tests, and "rgdal", bindings for GDAL.

The architecture of SDA system is graphically summarized in Figure 4.2 based on the selected FLOSS products. The spatial data display and data entry is done in QGIS environment. By this way, GIS software plays a role as the data integrator. Different spatial data formats can be overlaid in this environment with the help of GDAL that is used by both products; QGIS and R. The SDA routines are executed in R environment. The conversion of spatial data from GIS environment to statistical package or vice versa is done automatically at the background by using GDAL through Python. Displaying the output as a result of the analysis is also done in QGIS which means that GIS software can be used as visual evaluation tool. The SDA system has following several dependencies;

1. **QGIS:** FLOS GIS software as main GIS component of FLOS SDA system
2. **Qt:** Cross platform application development framework
3. **Python:** Programming languages and its modules; RPy2, and optional two modules; NumPy and Pywin32 required for Windows OS
4. **PyQt:** Python binding for Qt framework
5. **PyQGIS:** Python binding for QGIS library
6. **Two FLOS Spatial Libraries:** GDAL and PROJ.4
7. **R:** FLOS statistical package as the main SDA component of FLOS SDA system and its four external spatial packages; sp, maptools, spatstat and rgdal

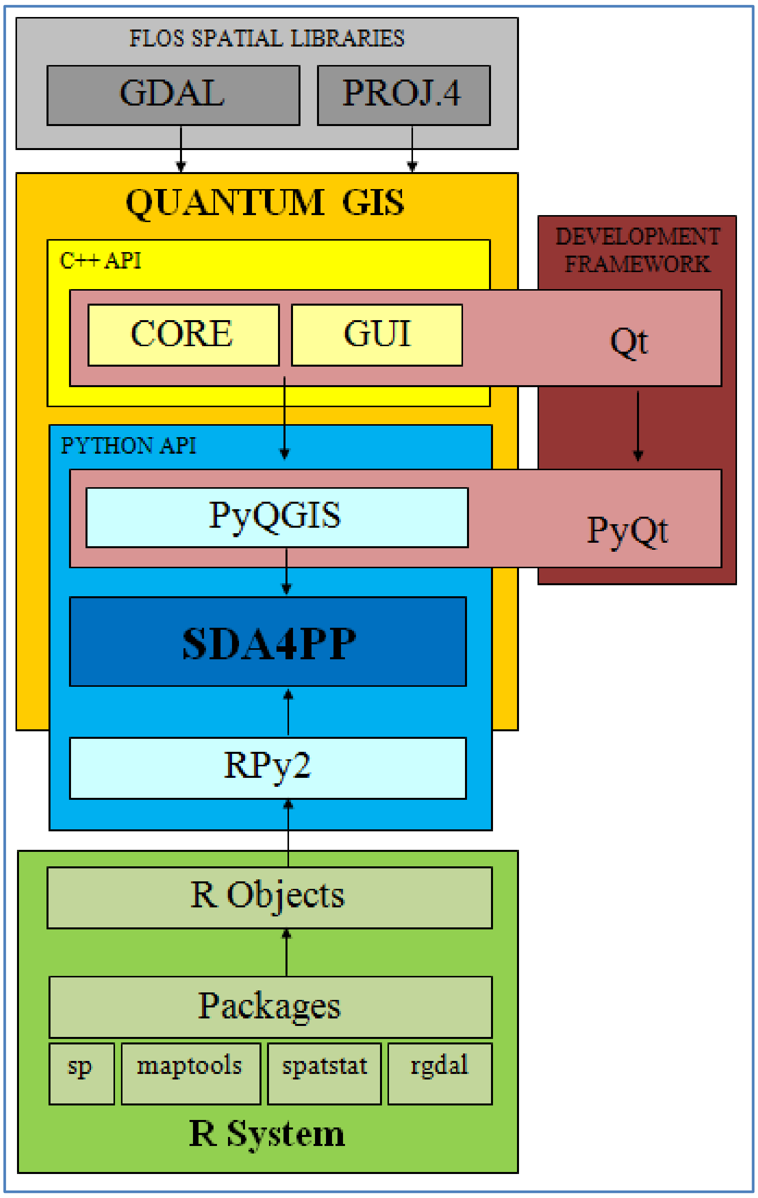


Figure 4. 2. Architectural Schema of Developed FLOS SDA System

SDA techniques are run in the R statistical packages, because there is no need to reinvent the algorithms of each SDA tool from the scratch, which is a very time consuming effort and will cause possible errors. This kind of work is out of the scope of the thesis. The disadvantage of R is that the processes are done with the command line interface, which is not user-friendly environment. It requires to have programming knowledge and is not appropriate for the GIS user who has no programming or statistical knowledge. Although R has many different packages for variety of fields, it is not GIS software and has got a different mentality or style of handling spatial data that is too much complicated and different for many researchers and GIS experts, who have become accustomed to use windows based user friendly GIS software.

One of the objectives of SDA system is to have cross platform support. SDA system should work in Windows, Linux and Mac OS X. There should be only one version of source code, not different versions for each platform. For this reason, which FLOSS product is the most appropriate for the realization of this integration, is widely researched at first in Linux operating system. QGIS, Qt, R and Python support cross platform. But it does not mean that the developer can develop such a code that will run also in different platforms. The developed source code should be tested under different platforms. In this thesis, the SDA system's source code is tested under Windows and Linux. There is no access to Mac OS X, so, this operating system is not tested. Theoretically, the system should also work in that OS. At first, the development stage has started in Ubuntu distribution version 7 Gutsy Gibson in January, 2008. After then, Windows XP Professional is used to develop the SDA system. At the final stage the system is working with the same source code at both operating system; Windows and Linux. The development stage of SDA system has been ongoing in both operating systems.

Each component is required to form SDA system. The properties of each component are explained in this section. The intention of this section is to review briefly the properties of each component. More detailed information about how the plug-in works with these components is given in Chapter 5.

4.5.1. Main GIS Component of FLOS SDA System: QGIS

QGIS is user friendly FLOS GIS software product which can be executed on Linux, UNIX, Mac OS X and Windows OS. QGIS uses GDAL library to read and write to many raster and vector formats. QGIS supports raster GIS data formats such as GeoTiff, Erdas Imagine Images, ArcInfo ASCII Grid, etc. and vector GIS data formats such as ESRI Shapefile, MapInfo, GML, etc. QGIS is licensed under GNU GPL. QGIS includes features such as connecting geographical databases like PostgreSQL by means of PostGIS plug-in and Spatialite database applications, overlaying data with different coordinate systems by automatically transforming data at the background (on the fly projection), creating and editing ESRI Shapefile format, etc. QGIS supports the Open Source Geospatial Consortium standards like web map service and web feature service. QGIS continues its activities under the umbrella of Open Source Geospatial Foundation. The development process of QGIS has started in May of 2002. Although this software is very new, the development process is on going fast due to having large, active and growing international community.

QGIS is developed by applying FLOSS development methodology. QGIS community serves to their users and developers with e-mail lists, forum, blog, wiki, preparing user guide, providing binary code with an installer and source code. QGIS is developed using the Qt framework with C++ programming language. This means that QGIS has a pleasing and easy to use GUI. Quantum GIS (2008) explains the aim of QGIS development as creating “easy to use GIS software which provides common functions and features”. Customized plug-in and GIS enabled applications using Python or C++ programming language can be created by accessing to the QGIS libraries.

QGIS has been designed in modular structure. Many features can be added to the software with plug-in. For example, one of core plug-in produces configuration file for MapServer application, which is powerful FLOSS developed for web GIS field. There are many QGIS core plug-in developed with either C++ or Python programming language. It is also possible to develop third party plug-in by the external developers who are not members of the community. All plug-in can be accessible from the repositories through the Internet. Apart from the project's official plug-in repository, external developers can publish their plug-in by creating plug-in repositories on their own web sites. For example, the plug-in repository of this thesis is published under the Department of Geodesy and Geographic Information Technologies in Middle East Technical University. This plug-in repository can be accessed from this web address; http://www.ggit.metu.edu.tr/~volkan/plugins.xml.

QGIS can be customized in two ways by using two programming languages. First method is to create a new GIS software, while the other one is to develop a new plug-in within QGIS. In the first method, the standard functions such as zoom in, zoom out, panning, querying, etc. required for generic GIS software can be added by calling related classes in QGIS library. The second method is to develop a new plug-in in the existing GUI of QGIS which includes all available tools provided by native interface of QGIS. This development can be carried out by anybody who knows how to develop in the QGIS. GNU GPL guarantees to develop a new software product by using existing source code of libraries and applications. QGIS can be developed with Python or C++ programming language. Development support with Python programming language has started in QGIS version 0.9.0 in 2008.

4.5.2. Cross Platform Application Development Framework: Qt

Trolltech (2008a) defines Qt as “the standard framework for developing high performance, cross platform applications and provides portability across many different operating systems”. The framework is capable of running on different operating systems with the same code. The framework is developed by Trolltech Company founded in 1994 in Oslo, Norway (Riverbank 2008a). Riverbank (2008a) defines this framework as Linux’s answer to Windows. It also offers a development environment that provides advanced and fast performance for C++ developers. The classes and methods in the framework are fully object oriented (Trolltech 2008a). Unlike some other GUI libraries, Qt was built using an object oriented design from the beginning (Riverbank 2008a).

History of Qt development environment dates back to ancient times in Linux world. Most applications developed in Linux world are written by use of two different application development frameworks; GTK and Qt (Vardar 2006). While GTK framework is used to develop Gnome desktop environment, the Qt framework constitutes the base of “K” development environment. Qt framework was first a fully paid product and then, the GNU GPL version has also been released (Vardar 2006). By this way, Qt has two licenses; GNU GPL version of Qt can be used in FLOSS projects, a license fee must be paid in proprietary projects (Riverbank 2008a). Qt commercial edition allows traditional commercial software distribution, whereas Qt open source edition is provided free of charge under Q public license and GNU GPL conditions (Trolltech 2008a). All Qt software development tools like Qt Designer, Qt Linguist, Qt Assistant and qmake are also included in open source edition and can be used free of charge to develop FLOSS products.

The Qt framework provides to programmers to develop a full functional application in many different application areas such as database, script, network, openGL, XML, multimedia, web applications, etc. The framework is also integrated with software development editors such as Visual Studio and Eclipse (Trolltech 2008c). Moreover, Qt framework develops and provides the Qt Designer which is a product that can design GUI visually (Riverbank 2008a). It also makes possible to develop with Java language by means of Jambi application (Trolltech 2008c). Corporations like Adobe, Google, Skype, Lucas, NASA, Walt Disney Feature Animation, IBM, Sharp, Siemens and Mathematica are well-known customers of Qt (Trolltech 2008b). Qt as a development environment had proven its power through its experience and the extensiveness of applications developed.

Application framework of QGIS is Qt that is naturally used while developing SDA system. QGIS can be used on both desktop environment and portable handheld devices because of Qt's capability to run on different platforms. This framework is not developed or specialized for GIS field. Therefore, other FLOS libraries and applications have been included within the QGIS project in order to have specific functionality such as managing geographic data, defining coordinate systems and realizing spatial applications specific to GIS field. Integrated FLOS libraries and applications in QGIS are Flex, Bison, GRASS, GEOS, GDAL, SQLITE, GSL, PROJ.4, PostGIS and Expat. These are the FLOSS products developed by other communities' members and used to enrich functionality of QGIS.

4.5.3. Programming Language: Python

Python is mostly known as an object oriented scripting language, although it is completely general purpose programming language. Actually, Python is a dynamic object oriented and high level programming language. Python programming language can be used for many purposes; web programming, database, GUI development, scientific applications, training, network, software and game development, etc. Main distinct capability of Python programming language is to integrate with other languages and installed with extensive standard libraries (Python 2008a). Python is commonly used as "glue" for other software components in the application (Lutz 2001). Python is being used by organizations and corporations such as NASA, Google, YouTube, Industrial Light and Magic, Firaxis Games, Infoseek, Yahoo, Red Hat, etc (Python 2008a). Main module of Python has been written with C programming language. Python runs fast due to shortness of the module and advantages brought by C. Subsequent coding have been carried out by Python language.

The name, Python, comes from the BBC comedy series named “Monty Python's Flying Circus”. Guido van Rossum, creator of Python programming language, has chosen this name since he enjoyed this comedy series so much. Rossum has started to develop Python at the end of 1989. Rossum has decided to develop a new programming language since ABC language in the Institute could not be extended and he needed to create a better method instead of C or Bourne shell scripts for the operating system (Deitel et al. 2002). His aim is to create a new language that includes data types at very high level and capable of interfering at system level. Therefore, Python language includes the features of C, C++, ABC, Modula-3, Icon and other languages (Lutz 2001). Also, it will have mobility at system level as C and capability to be used with different languages. Python language is designed at the beginning to be used as extensible language. Also, all source code of Python interpreter is open which makes to possible to embed the interpreter in any application and can be used as script language of that application.

License of Python has been adapted to GNU GPL and its use is not subject to any limitation. It can be used free of charge even for commercial purpose. Source code of Python and installation files can be obtained from the official website of Python. Python language was announced at USENET on February 1991 (Python 2008b). Python is nearly 20 years old sufficiently stable, mature, powerful and flexible programming language. It is advised as the first programming language to be learned because of its ease of learning. The number of Pythonists (Python developers and users) even cannot be estimated, since Python comes as default installed language in many Linux distributions such as Fedora, Debian, Pardus, etc. TIOBE (2010) provides a programming community index which is used to determine the popularity of programming languages. This index ranks the programming languages according to their popularities as shown in Table 4.3. Python position was at 7th rank in August 2010, 8th rank in 2005, and 24th rank in 1995. Popularity of Python programming language is increasing in time.

Table 4. 3. Position of Top 10 Programming Languages for 5, 15 and 25 Years Ago (TIOBE 2010)

Programming Language	**Position**			
	Aug 2010	**Aug 2005**	**Aug 1995**	**Aug 1985**
Java	1	1	-	-
C	2	2	3	1
C++	3	3	2	11
PHP	4	5	-	-
(Visual) Basic	5	6	1	4
C#	6	7	-	-
Python	7	8	24	-
Perl	8	4	8	-
Objective-C	9	43	-	-
Delphi	10	10	-	-
Lisp/Scheme/Clojure	16	14	6	2
Ada	29	17	7	3

Since Python is a FLOS programming language, it is possible to learn how it runs by examination of its source code. Python can be used in two ways. Firstly, the applications and libraries written with different languages such as C, C++, FORTRAN, Java (with Jython) and .Net (with IronPython) can be called with Python language. This extendibility feature is one of the most distinctive aims of Python programming language. This feature makes Python so powerful programming language that it has the ability to take existing libraries, written in C or C++ programming language.

The second way is to embed Python interpreter in another application. Benefit of this process is the capability to use Python utilities and libraries with another language within an application. The programmer can use Python as a script language by embedding it into the applications (Brown 2001). SIP and SWIG are two FLOSS examples that simplify work of programmer to use C or C++ libraries with Python. SWIG is one of the most frequently used software for extendibility. SIP is developed by Riverbank Computing Limited to provide access to Qt framework with Python. SIP is specifically designed for bringing together Python and C/C++. SIP was originally developed to create PyQt, the Python bindings for the Qt framework, but it can be used to create bindings for any C or C++ library.

There are many applications that are developed with Python. One example can be given from Turkey. For example, Pardus distribution is developed by Python programming language in TUBITAK National Electronics and Cryptology Research Institute. Python language has started to become increasingly popular in Turkey. Another example is from GIS field. Script language of ESRI ArcGIS, proprietary desktop GIS software, has become Python. ESRI has shifted from using Arc Macro Language in their products to Python as script language starting from ArcGIS version 9.0 since May 2004 due to the potential and capability of Python. Another example is Thuban that is the example from FLOS GIS software which is developed only with Python programming language. It has also become possible to develop QGIS software with Python programming languages with the release of version 0.9.0. QGIS integrated the expandable and embeddable properties of Python.

Python programming syntax does not require use of a lot of different punctuation marks frequently that make difficult to understand the source code. Python’s syntax is easy to read, since its language structure is very close to English. It allows carrying out more tasks by much less code with respect to C++. It provides garbage collection tools and various GUI development environments like Java. It comes with an extensive development library. Codes can be executed without compilation. Python can run on many different operating systems. In fact, the programs developed with Python run today in every notebook, personal computers and handheld personal digital assistants (Lutz 2001).

Since Python is an interpreted programming language, it can run a little bit slower with respect to compiled languages like C/C++ programming language. This performance constraint can be important for the ones dealing with large amounts of data, expecting fast performance and planning to use it in embedded systems. This disadvantage may not create a huge performance difference due to today's powerful computers. Python can be a preferred programming language with respect to C/C++ due to software development utilities and speed provided to programmers. The slowness is related to running performance of the program. Python modules are recurrently tested to provide best performance and continuously checked by thousands of programmers. It cannot be sure to achieve more performance by a code written in C only with one developer with respect to Python modules against this accumulation of experience.

To overcome the performance constraint, the code can be developed quickly with Python like producing prototype. Complicated and advance algorithms and applications can be written quickly first in Python and then they can be rewritten with C programming language. Principles of agile application development method can be easily realized by the use of Python language. Today, when high processing speeds of computers are considered, it is more logical to develop software in a programming language which gives priority to performance of the programmer instead of working performance of machines.

Python is an interpreted programming language. Python does not need compilation to binary (Swaroop 2005). Brown (2001) explains the interpretation process as that "Python source code is always translated into a byte code representation, and then, this code is executed by the Python virtual machine". This process is faster in Python, because this byte code is written into a file whose extension is "PYC". If there is no change in the code for the module stored as in the file, that file is not parsed. The use of both interpretation and byte code stages helps to improve performance of Python. As compared with pure interpreters such as BASIC, Python is faster, but slower when compared with compiled languages such as C and Pascal (Brown 2001). All this makes using Python much easier, because the programmer does not have to wait to compile the program (Swaroop 2005). While programming with Python, there is no such a waiting time for compilation, because Python is interpreted language. Another time saving advantage of Python is to have interpreter shell. This shell is useful especially if the programmer wants to test the code quickly. The programmer can easily check the code in the interpreter shell and get the results immediately before making real changes in the program. This property also decreases the consumed time during the development of application. The benefit of interpreted and high level languages is much more valuable when compared with the computation speed. At least half of the development schedule would be gain with Python as compared with C programming language (Brown 2001).

Another benefit of using Python programming language is that Python solution is about half size of a comparable C++ solution, which means that more functionality can be obtained with less code, causing more maintainable software (Riverbank

2008a). Langtangen (2006) mentions that “Python enables to write programs that are significantly shorter code than programs written with FORTRAN, C, C++ or Java”. Another shortest code comparison between Python and C# code is given as an example in Figure 4.3. These two codes do the same work. The code opens the file, read the text inside of the file, changes all texts to upper case and displays the upper cased text in the screen. C# code has 219 characters without space, whereas Python code is 38 characters. Python code is nearly six times smaller than C#. Lutz (2001) summarizes the advantageous of using Python programming language for developers as that “the system can be implemented quickly and the resulting system will be maintainable, portable, and easily integrated with other application components”.

C# Code	**Python Code**

using System;
using System.IO;
class Hello { static void Main (){FileStream file = new FileStream ("sample.txt", FileMode.OpenOrCreate, FileAccess.Read);
    StreamReader sr = new StreamReader (file);
    Console.WriteLine (sr.ReadToEnd ().ToUpper ());
} }
	Print open ('sample.txt').read ().upper ()

Figure 4. 3. The Length of Code Comparison between Python and C# Programming Languages (Aktaş 2010)

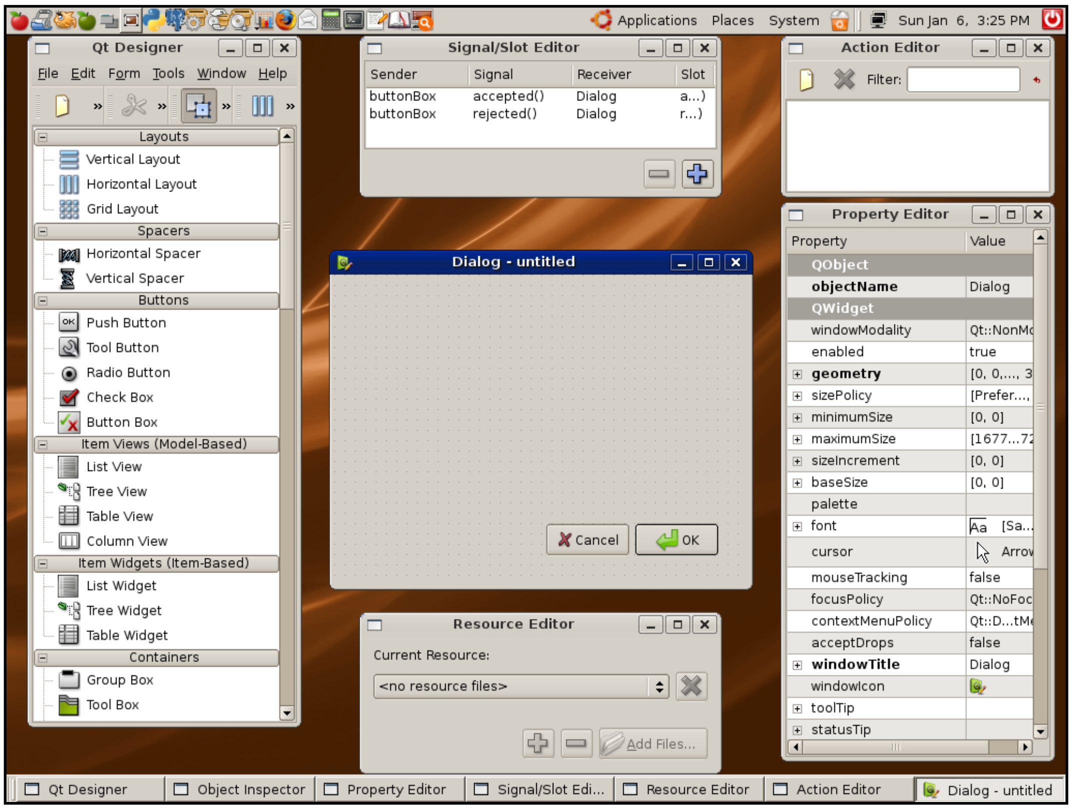
Python developers can find many FLOS GUI toolkits. Tkinter is the default GUI toolkit for Python (Langtangen 2006). PyGtk, PyQt and wxPython are the most known and used other FLOS Python GUI examples. These GUI examples are the Python bindings for Gtk, Qt and wxWindows libraries, respectively (Lutz 2001). Langtangen (2006) states that Tkinter is easier GUI toolkit than PyGtk, PyQt, and wxPython, however, PyGtk, PyQt, and wxPython have become more popular than Tkinter. These libraries are developed with C/C++ and should be installed in the system in order to use them with Python (Langtangen 2006). Swaroop (2005) defines the term “Python binding” as an ability to use the libraries with Python which are written in C/C++ or other languages. Langtangen (2006) defines two ways for developing GUI; “either the developer writes a Python program calling up functionality in the GUI toolkit, or applies a graphical designer tool to design the GUI interactively and visually on the screen”. For example, Qt Designer is such a graphical designer tool in the framework of Qt. The GUI file of Qt Designer can be converted to Python syntax with the help of PyQt tools.

Advantage of high level programming language such as Java and Python is to perform more tasks with less code due to extensive library support. It is more advantageous to use a high level programming language such as Python in order to quickly and easily develop a product like QGIS which utilizes many libraries. Therefore, Python programming language has been chosen instead of C++ to develop QGIS.

4.5.4. Python Development with Qt Framework: PyQt

PyQt is a binding library which provides access to C++ libraries of Qt application development framework by use of Python programming language (Riverbank 2008a). PyQt is being developed by Riverbank Computing Limited Company. This company is specialized in FLOS software technologies. PyQt development environment allows access to more than 600 classes in Qt library by using Python language (Riverbank 2008a). PyQt combines all potentials and advantages of Qt framework and Python language. Thus, developers find an opportunity to use power of Qt and simplicity of Python language together. PyQt is ideal to develop powerful, flexible and fast GUI applications. PyQt runs on all operating systems on which Python and Qt can be installed. It supports all Windows editions, most of Unix and its derivatives running with X11, Mac OS X, Linux, Solaris and HP-UX operating systems (Riverbank 2008a). General appearances of products developed by PyQt are in harmony with self appearances of aforementioned operating systems. In background, PyQt itself carries all processes needed to provide this harmony without being a burden to programmer (Riverbank 2008a). Python is suitable for fast software development method due to its capability of using C or C++ libraries.

PyQt developer can build up entire program by coding and also the developer can use the tools provided by Qt framework. For example, Qt Designer attached in the Qt framework designs GUI visually (Figure 4.4) and produces a project file in XML structure (Riverbank 2008a). One of PyQt tools can convert this XML file to Python code. The same process can be applied to resource collection file as well as translation file. While icons and images are described in a resource collection file in the XML structure for Qt Designer application, the resource collection file can be converted to Python code with another PyQt tool. For example, PyQt tools such as pyrcc4, pyuic4 and pylupdate4 can be used to convert the collection, GUI and translation file to Python code, respectively (Riverbank 2008b).



Sender	Signal	Receiver	Slot
buttonBox	accepted()	Dialog	a...)
buttonBox	rejected()	Dialog	r...)

Property	Value
QObject	
objectName	Dialog
QWidget	
windowModality	Qt::NonMo
enabled	true
geometry	[0, 0,..., 3
sizePolicy	[Prefer...,
minimumSize	[0, 0]
maximumSize	[1677...72
sizeIncrement	[0, 0]
baseSize	[0, 0]
palette	
font	Aa [Sa...
cursor	Arrov
mouseTracking	false
focusPolicy	Qt::NoFoc
contextMenuPolicy	Qt::D...tMe
acceptDrops	false
windowTitle	Dialog
windowIcon	
toolTip	
statusTip	

Figure 4. 4. Snapshot of Qt Designer in Ubuntu Distribution

PyQt is also dual licensed like Qt. PyQt is licensed under GNU GPL for FLOSS projects and commercial software license for proprietary software projects (Riverbank 2008a). In other words, if proprietary software will be developed by using Python language with Qt framework, since PyQt is necessary, it is also required to pay to both developers of Qt and PyQt (Riverbank 2008b).

In fact, there are many GUI libraries available for using with Python programming language. According to Rempt (2001), these GUI libraries like Tkinter, PyQt and wxPython are the really serious options for the programmer. His personnel choice is PyQt due to based on criteria of performance, programming model, rich functionality of the widgets and ease of installation. Rempt (2001) explains the most important features of PyQt as follows:

- *Based on Qt framework developed with C++ programming language*
- *Runs on different operating systems with the same code*
- *Uses signal and slot mechanism to couple GUI items with event driven actions*
- *Binds almost the complete Qt library*
- *Allows sub classing of Qt classes in Python*
- *Allows applications to look like native feel of operating system*
- *Comes with full rich of advanced GUI controls*

In the development of SDA system, PyQt is also used as the Python GUI toolkit in QGIS. However, this choice is not deliberately done according to either the features of PyQt or Rempt's criterion. PyQt is applied in the development, because initial development of QGIS has started with Qt framework. Instead of trying to use another GUI toolkit, Qt framework is accepted by default as the main GUI toolkit for SDA system in order to provide seamless integration between the developed system and the QGIS. Probably, Gary Sherman, who is the founder of the QGIS, has been impressed from the features of Qt, while deciding to choose this framework for the development of QGIS software.

4.5.5. QGIS Development Environment with Python: PyQGIS

The main development environment of QGIS is Qt framework. PyQt is the Python binding for Qt libraries. The classes in the Qt framework written with C++ programming language can be called, accessed and used by Python commands due to extendibility feature of Python language. Usage of Qt libraries with Python language has become possible by Qt version 4.2 since October 2006. PyQt feature has been added to QGIS library with version 0.9.0 since 2008. Since then, it has become possible to develop the QGIS software with Python language. QGIS library is continuously binded with SIP tool in order to be called with Python language. The structure of Python QGIS Application Programmers Interface (API) is 99% same with the structure of C++ QGIS API.

PyQGIS is created by linking QGIS library to Python language with the use of SIP tool. For this binding task, the SIP, one of FLOSS applications, is preferred instead of SWIG application, which is more popular and broadly used program than SIP. SIP tool has been preferred, since PyQt has been binded to Qt by use of SIP tool. While

Qt is converted to PyQt by SIP tool, QGIS is also converted to PyQGIS by the same tool in order to continue a solid integration and provide a complete compatibility (Dobias 2008). By this way, both QGIS and Qt libraries are brought together at Python programming language. This binding makes possible for Python language to develop new GIS application by using QGIS library, to perform certain tasks automatically within QGIS and to create plug-in within the main window of existing QGIS interface.

Python is also embedded in the QGIS as an interpreter which interprets Python plug-in source code at the start up of the QGIS. Python interpreter can also be accessed and used with integrated console in the main interface of QGIS. By means of using this console, the programmers can see the results of Python commands by testing them in QGIS (Dobias 2006). Python console should exist in the "Plug-in" menu of QGIS. If this command is not available, it means that Python support is not installed. Python is required to install to have this console. With the help of this binding, either a new plug-in or a new GIS application can be developed by using QGIS and Qt libraries with Python language.

4.5.6. FLOS Spatial Libraries: GDAL and PROJ.4

Geospatial Data Abstraction Library (GDAL) is one of implemented FLOS spatial libraries used for converting both raster and vector GIS data formats in SDA system. OSGEO (2011) defines GDAL as "a cross platform C++ translator library for raster and vector GIS data formats". The license of this library is Massachusetts Institute of Technology, one of open source licenses. This license makes possible to use the library even with proprietary software. The development of this library is continuing under the umbrella of Open Source Geospatial Foundation. A variety of useful command line utilities are available for data translation (Warmerdam 2008). OGR Library is a C++ FLOS library used for data translation of vector GIS data format. Mitchell (2005) explains that "OGR library is part of the GDAL Project and is packaged with GDAL". The library has also command line tools, which provide for reading and writing to a variety of vector file formats. For example, ESRI Shapefile, S-57, SDTS, PostGIS, Oracle Spatial, and MapInfo MID/MIF and TAB formats are some of the supported vector GIS data formats in OGR library (GDAL 2011d). The GDAL project was launched in 1998 by Frank Warmerdam and includes 20 contributing developers in 2008 (Warmerdam 2008).

GDAL deals with raster data, whereas OGR deals with vector data. Common operations on raster formats such as warping, converting, merging, and applying coordinate transformation are also provided by command line utilities of GDAL (Sherman 2008). GDAL supports 118 raster formats and OGR supports 56 vector formats as in February 9, 2011. Sherman (2008) lists some of the popular raster formats supported by GDAL such as ArcInfo ASCII Grid, ArcInfo Binary Grid (.adf), First Generation USGS DOQ (.doq), New Labelled USGS DOQ (.doq), ERMapper Compressed Wavelets (.ecw), ESRI .hdr labelled, ENVI .hdr labelled Raster, GMT Compatible netCDF, GRASS Rasters, TIFF/GeoTiff (.tif), GXF – Grid eXchange File, Hierarchical Data Format Release 4 and 5 (HDF4 and 5), Erdas Imagine (.img), JPEG, JFIF (.jpg), JPEG2000 (.jp2, .j2k), MrSID, NetCDF, Portable Network Graphics (.png), ArcSDE Raster, USGS SDTS DEM (*CATD.DDF), and USGS ASCII DEM (.dem). OGR provides tools to manipulate vector GIS layers (Sherman 2008). In some cases, the OGR library can create and write to the formats, while some of the vector data formats are supported as read-only. Mitchell (2005) lists the vector data formats supported by OGR such as ArcInfo Binary Coverage, ESRI Shapefile, DODS/OPeNDAP, FMEObjects Gateway, GML, IHO S-57 (ENC), MapInfo, MicroStation DGN, OGDI vectors, ODBC, Oracle Spatial, PostgreSQL, SDTS, UK .NTF, US Census TIGER/Line and Virtual Datasource (VRT).

GDAL runs on all modern flavours of UNIX; Linux, Solaris, Mac OS X, and most versions of Microsoft Windows (GDAL 2011b). GDAL library can be directly linked which helps to be included both in proprietary and FLOS GIS software products (Mitchell 2005). OSGEO (2011) explains that GDAL is used as “data access engine for many applications including MapServer, GRASS, QGIS, and OpenEV”. It is also used to varying degrees by a variety of proprietary software products such as FME, ESRI ArcGIS, and Cadcorp SIS (Warmerdam 2008). GDAL is considered as one of major projects used in both FLOS and commercial GIS community due to comprehensive set of functionalities. GDAL is written in ANSI C and C++ and can be compiled with all modern C/C++ compilers (GDAL 2011b).

Mitchell (2005) explains the most three important features of GDAL library; firstly, the library supports dozen of raster and vector formats; secondly, it is available for other applications to use; thirdly, prebuilt utilities help the users to use the functionality of the GDAL library without writing any code. The bindings of GDAL in other languages such as Perl, Python, Visual Basic, R, Ruby, Java and C-Sharp are also available (GDAL 2011a). This is accomplished using the SWIG utility (Warmerdam 2008). Python bindings have a substantial user base (Warmerdam 2008). Python was the first set of bindings supported by GDAL. Although the bindings were generated with SWIG, the process was very Python specific and contained a significant amount of hand written wrapped code (GDAL 2011c).

Another implemented FLOS geospatial library in development of SDA system is PROJ.4. It is a cartographic projections library that is used in many FLOS GIS software products (Sherman 2008). Since May 2008, the project has become part of the MetaCRS project, which tries to organize projection related libraries (PROJ4 2011a). MetaCRS is in the incubation stage in the Open Source Geospatial Foundation in February 2011. PROJ.4 was originally developed by the US Geological Survey and is now maintained by a group of volunteers (Sherman 2008). PROJ.4 is actively used by number of FLOSS projects such as GRASS GIS, MapServer, PostGIS, Thuban, OGDI, Mapnik, TopoCad, etc. (PROJ4 2011a). Datum shifts can be approximated with 3 and 7 parameter transformations by means of PROJ.4 library (PROJ4 2011b). Conversion between EPSG and PROJ.4 format is also possible (PROJ4 2011a). The library also comes with some command line utilities for experimenting with projections and doing interactive transformations (Sherman 2008). It is released under a Massachusetts Institute of Technology license, which allows to make proprietary derivatives (PROJ4 2011a).

4.5.7. FLOS Statistical Software: R and Spatial Packages

R is an environment and programming language for data analysis and graphics (Everitt and Hothorn 2006). R can run on many different operating systems such as UNIX, Windows and Mac OS (R-CRAN 2008b). The base distribution of R and user contributed packages are licensed under GNU GPL (Everitt and Hothorn 2006). A core R development team is formed in 1997 (R-CRAN 2008a). This team is formed by a small group of statisticians who can modify the source code and maintains the base distribution of R. Much functionality is implemented by user contributed packages as add-on. These add-on packages are authored and maintained by a large group of volunteers (Everitt and Hothorn 2006). The packages are developed for different variety of specific purposes (R-CRAN 2008a). The complete source code of add-on packages and the base distribution of R can be accessible through the Internet. The implementation of special methods can be investigated from the source codes of add-on packages through Internet (Everitt and Hothorn 2006).

Add-on packages of R are mainly formed in four ways; base distribution of R, Comprehensive R Archive Network (CRAN), Omegahat Project and Bioconductor Project. The number of packages was 12 in base distribution of R, 1,451 in CRAN, 46 in Omegahat Project, and 244 packages in Bioconductor Project in May 2008. The Omegahat Project provides a variety of FLOS software for statistical applications (R-CRAN 2008b). CRAN contains great number of user contributed add-on packages. The number of user contributed add-on packages in CRAN is growing day by day. Each package may depend on several other packages. In order to install one package, each dependency should also be installed. The Bioconductor Project produces an FLOS software framework for biologists and statisticians working in bioinformatics (R-CRAN 2008b).

Although R has dedicated data structures for specialized methods, the similar approach is not valid for spatial data. There are a lot of R packages dealing with spatial statistical methods and GIS data formats, but, there is little coordination between them (Bivand 2007). It is trying to organize the structure of spatial data in one package which is called as “sp” package which is based on S4 classes. The sp package will become the base package for other spatial packages. While dealing with spatial packages in R, it is advised to use the structure of sp package for handling spatial data.

The SDA system uses mainly four packages; namely, sp, maptools, spatstat and rgdal. The implementation is focused on point pattern analysis. Spatstat package is used for SDA. Maptools is a package that contains tools for converting and handling spatial objects. This package is used to handle projection system and conversion between S4 classes of sp package and ppp format of spatstat package. Rgdal package is the R binding for GDAL. Rgdal package is used to read and write vector GIS data format like ESRI Shapefile and MapInfo, and raster GIS data format like georeferenced TIFF image file and ERDAS Imagine Images.

R is binded with Python by using RPy2, one of Python module. Another Python module binding R is RSPython which is developed in the Omegahat project. RPy2 Python module is used instead of RSPython module, because it is more robust and has easy interface for using R from Python (RPy 2008). All R objects and functions available in add-on packages and the base distribution of R can be executed with Python language through RPy2 module. By this way, R language's errors can also be handled with Python code.

CHAPTER 5

IMPLEMENTATION AS SDA4PP PLUG-IN

5.1. Development of Python Plug-in in QGIS

Spatial Data Analysis for Point Pattern (SDA4PP) plug-in is the outcome of development of FLOS SDA system. Development of SDA4PP plug-in started in January 2008 and first announcement was done to QGIS community in September 2008. During the development of system, all revisions of source codes of the plug-in are published in the QGIS Python repository which can be accessed any time through Internet connection to download and install the plug-in. During testing and debugging by the user and member of QGIS community, several bugs are found and reported to the author of this thesis via e-mail. These bugs are corrected as soon as possible and revised versions are distributed as well in the repository. Development of plug-in is completed in 2.5 years between January 2008 and June 2010. During this period, the plug-in is revised 196 times as 5 major releases in every 6 month periods.

Selection of integrated SDA tools is done according to the definitions of Bailey and Gatrell, and the availability of techniques in the spatstat package. SDA field has wide range of different analysis techniques. In order to ease the development of SDA system, the integrated tools are limited according to the point pattern analysis. Graphical user interface of integrated SDA tools and their analysis results are given in Appendix C. Integrated tools can be classified into four headings; SDA, exploratory SDA, GIS and Utility tools. Following tools are developed in the plug-in by using Python programming language:

A. SDA Tools

1. **Uniform Intensity:** Displays intensity value and useful summary of a point pattern dataset.

2. **Kernel Smoothed Density:** Computes a kernel smoothed intensity function from a point pattern.
3. **Adaptive Density:** Computes an adaptive estimate of the intensity function of a point pattern.
4. **G and F Estimate Functions:** G function estimates the nearest neighbour distance distribution function G(r) and F function estimates the empty space function F(r) from a point pattern in a window of arbitrary shape.
5. **Ripley's K and L Functions:** K function estimates Ripley's reduced second moment function K(r) from a point pattern in a window of arbitrary shape and L function calculates an estimate of Ripley's L-function for a spatial point pattern.
6. **Local Ripley's K and L:** Computes the neighbourhood density function, a local version of the K-function defined by Getis and Franklin in 1987.
7. **Quadrat Test:** Performs a chi-squared test of Complete Spatial Randomness for a given point pattern, based on quadrat counts. Alternatively the tool performs a chi-squared goodness-of-fit test of a fitted inhomogeneous Poisson model.
8. **Kolmogorov-Smirnov Test**: Performs a Kolmogorov-Smirnov test of goodness-of-fit of a Poisson point process model. The test compares the observed and predicted distributions of the values of a spatial covariate.
9. **Simulation Envelope of CSR**: Computes simulation envelopes of a summary function for complete spatial randomness.
10. **Fit Poisson Model:** Creates an instance of the Poisson point process model which can then be fitted to point pattern data.
11. **Kriging:** Performs automatic kriging on the given dataset. The variogram is generated automatically using autofitVariogram function.

B. Exploratory SDA Tools

1. **Rggobi:** Bindings of GGobi which is an FLOS visualization program for exploring high-dimensional data. It provides highly dynamic and interactive graphics such as tours, as well as familiar graphics such as the scatterplot, barchart and parallel coordinates plots. Plots are interactive and linked with brushing and identification.
2. **Linked Statistical Display:** Provides high interaction statistical graphics. It offers a wide variety of plots, including histograms, barcharts,

scatterplots, boxplots, fluctuation diagrams, parallel coordinates plots and spineplots. All plots support interactive features, such as querying, linked highlighting, color brushing, and interactive changing of parameters.

3. **Interactive Identify:** Gives customizeable feature information for all overlapping raster and vector layers at the same time in one window.

C. GIS Tools

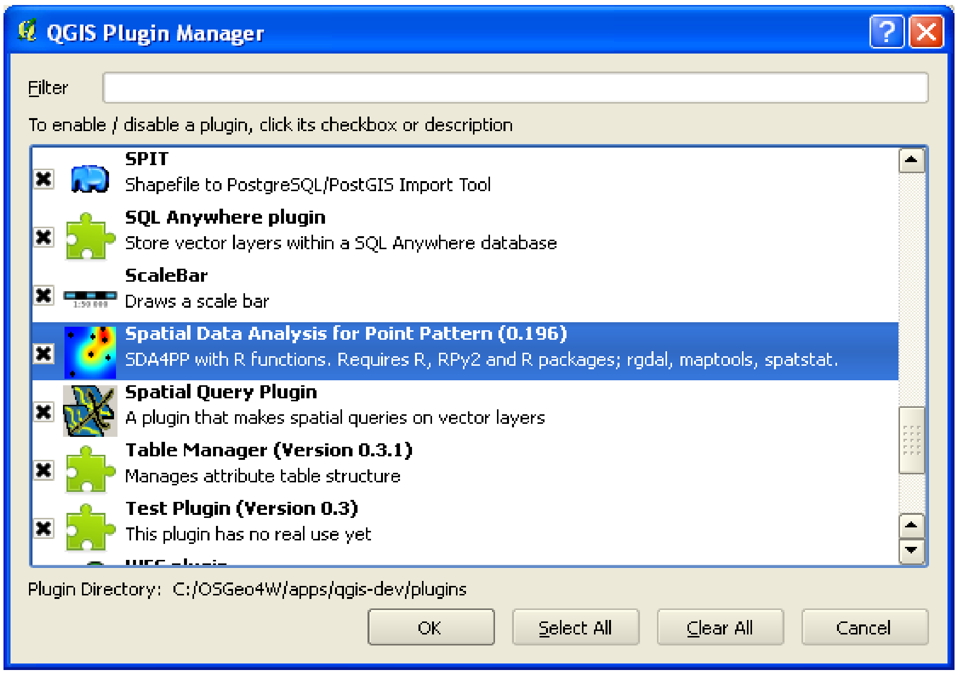
1. **Generate Random Point:** Generates random point according to two Poisson Processes; (In)Homogeneous Poisson Process generates a random point pattern using the (homogeneous or inhomogeneous) Poisson process, and Gauss Poisson Process generates a random point pattern, a simulated realisation of the Gauss-Poisson Process.
2. **Generate Regular Point:** Generates regular point according to either distance between points or the number of point in a specified extent of layer.
3. **Nearest Point and Distance:** Finds the nearest neighbour of each point, computes the distance from each point to its nearest neighbour and creates the matrix of distances between all pairs of points in a point pattern. The tool also finds the nearest neighbour in Y of each point of X in given two point patterns X and Y.
4. **Polygon to Point (Centroid):** Computes the centroid of polygon feature and saves the computation as point layer.
5. **Point to Polygon (Delaunay or Dirichlet):** Computes the Dirichlet/Voronoi tessellation or the Delaunay triangulation of a spatial point pattern.

D. Utility Tools

1. **R Console:** Provides a console to R environment thorugh RPy2 Python module. R code syntax is required to call R objects.
2. **Graphic Display Option:** Setups the settings for native R graphic device
3. **Components/Install Missing R Packages:** Checks the components of the system and if a component is missing, the dialog displays the information about the missing component. If not, continues to check R packages and if missing, the dialog gives options for users to download missing packages from the Internet and installs them.

The structure of QGIS plug-in is explained in the QGIS user guide (Quantum GIS 2008). New features can be added to QGIS with plug-in architecture that are two types of plug-in; core and user contributed plug-in. Core plug-in is developed by QGIS team and user contributed one by external developers. QGIS is distributed with entire core plug-in, whereas user contributed plug-in is in source form. Compilation of user contributed plug-in is not required for the plug-in developed with Python, but required for C++ programming language. While QGIS is opening, the system interprets the Python plug-in code. If any error is found in the plug-in, the system specifies the error and its line with the name of source file where the error occurred, and disables the plug-in. This plug-in could not be used until the error is fixed. The source code of Python plug-in is available under QGIS Python plug-in directory, so that, the developer has a chance to fix the error. The binary code of Python plug-in can also be installed under this directory. Although it is against the GNU GPL, there is no mechanism for checking not to use the binary code of Python plug-in in QGIS. Distributing the source code of plug-in is the social responsibility of the programmer who is developing plug-in licensed with GNU GPL in the QGIS environment.

All plug-in (the core and the user contributed plug-in) can be activated in the plug-in manager. The plug-in manager can be reached from the tools menu in QGIS. When the plug-in is checked in the manager window, it is loaded and becomes ready to be used in the QGIS. The plug-in manager dialog displays all installed plug-in as shown in Figure 5.1. When a listed plug-in in the plug-in manager is checked, the plug-in becomes ready to use in the main window of QGIS.



QGIS Plugin Manager

Filter

To enable / disable a plugin, click its checkbox or description

☒ **SPIT**
Shapefile to PostgreSQL/PostGIS Import Tool

☒ **SQL Anywhere plugin**
Store vector layers within a SQL Anywhere database

☒ **ScaleBar**
Draws a scale bar

☒ **Spatial Data Analysis for Point Pattern (0.196)**
SDA4PP with R functions. Requires R, RPy2 and R packages; rgdal, maptools, spatstat.

☒ **Spatial Query Plugin**
A plugin that makes spatial queries on vector layers

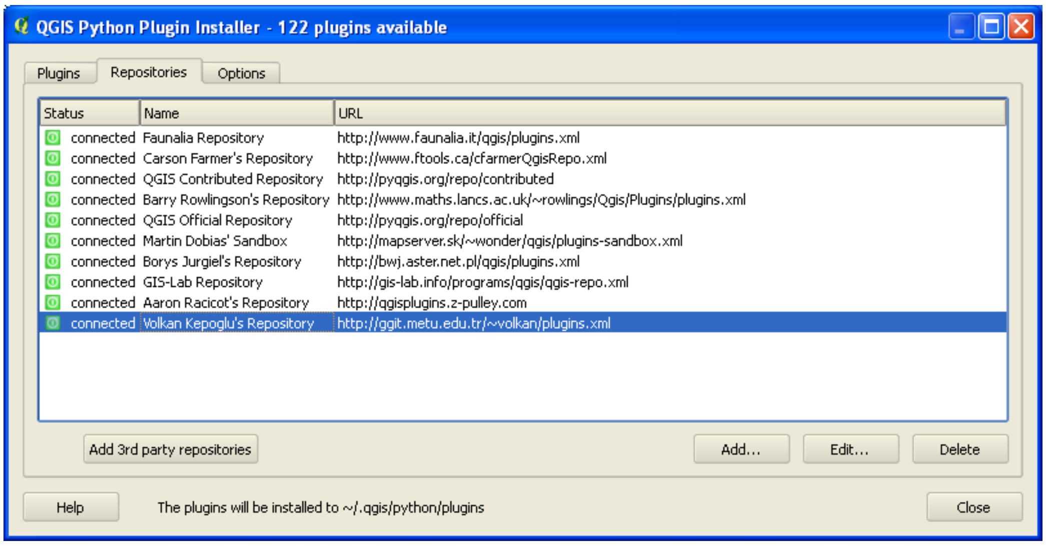
☒ **Table Manager (Version 0.3.1)**
Manages attribute table structure

☒ **Test Plugin (Version 0.3)**
This plugin has no real use yet

Plugin Directory: C:/OSGeo4W/apps/qgis-dev/plugins

Figure 5. 1. Snapshot of Plug-in Manager in QGIS Version 1.7.0 in March 2011

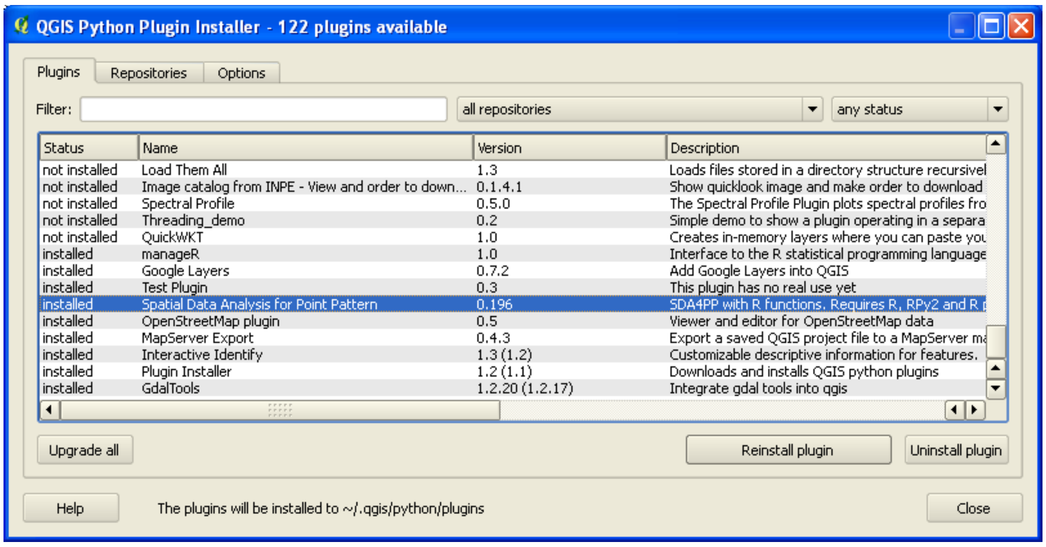
External developer's plug-in developed with Python can be installed by QGIS plug-in installer. The QGIS Python plug-in installer communicates with the repositories through the Internet connection which first lists available all repositories (Figure 5.2) and then, the plug-in in these repositories (Figure 5.3). When the user prefers to install one of the plug-in, the installer downloads the source code of plug-in as a zip file under the specified plug-in path and extracts the zip file. When QGIS is restarted, it recognizes this newly installed plug-in and has become ready to activate in the QGIS plug-in manager by user in order to use it in the QGIS. Downloading from the Internet and interpreting the source code of the plug-in use the QGIS users' credits. This linkage mechanism could be dangerous from security point of view, because it requires downloading and running the source code from the repositories owned by external developers.


QGIS Python Plugin Installer - 122 plugins available
Plugins
Repositories
Options
Add 3rd party repositories
Add...
Edit...
Delete
Help
Close

Status	Name	URL
connected	Faunalia Repository	http://www.faunalia.it/qgis/plugins.xml
connected	Carson Farmer's Repository	http://www.ftools.ca/cfarmerQgisRepo.xml
connected	QGIS Contributed Repository	http://pyqgis.org/repo/contributed
connected	Barry Rowlingson's Repository	http://www.maths.lancs.ac.uk/~rowlings/Qgis/Plugins/plugins.xml
connected	QGIS Official Repository	http://pyqgis.org/repo/official
connected	Martin Dobias' Sandbox	http://mapserver.sk/~wonder/qgis/plugins-sandbox.xml
connected	Borys Jurgiel's Repository	http://bwj.aster.net.pl/qgis/plugins.xml
connected	GIS-Lab Repository	http://gis-lab.info/programs/qgis/qgis-repo.xml
connected	Aaron Racicot's Repository	http://qgisplugins.z-pulley.com
connected	Volkan Kepoglu's Repository	http://ggit.metu.edu.tr/~volkan/plugins.xml

The plugins will be installed to ~/.qgis/python/plugins

Figure 5. 2. Available External Authors' Python Plug-in Repositories (Snapshot of Repositories Tab in Python Plug-in Installer in QGIS Version 1.7.0 in March 2011)



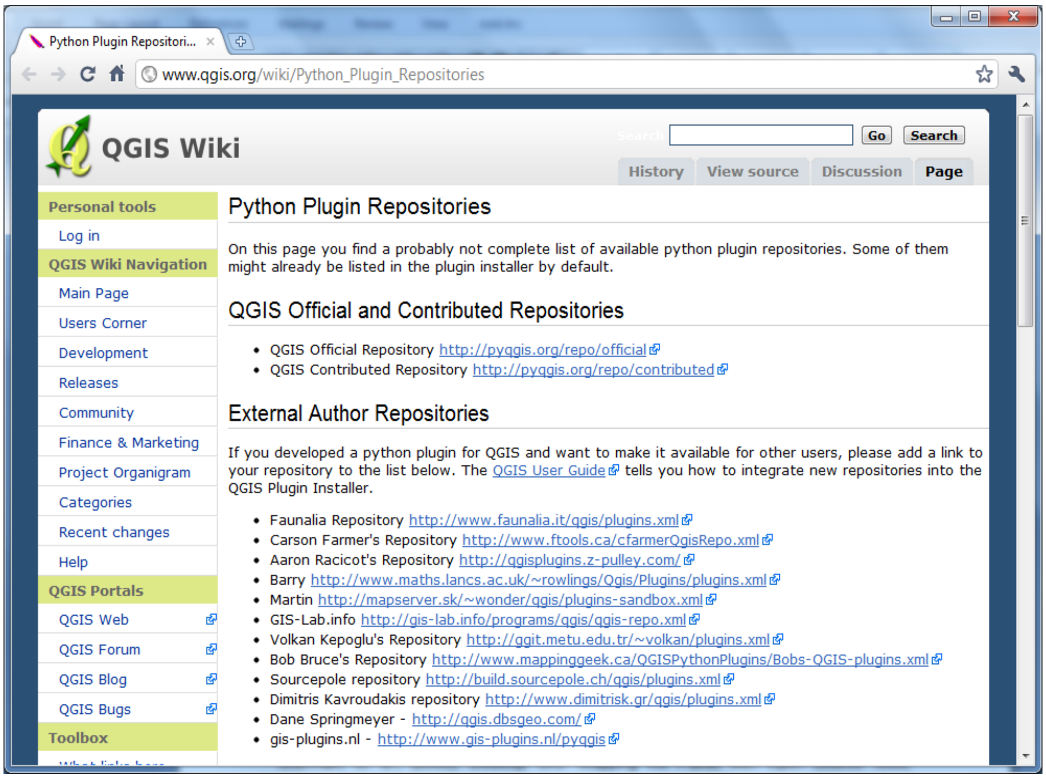
Filter: all repositories any status

Status	Name	Version	Description
not installed	Load Them All	1.3	Loads files stored in a directory structure recursivel
not installed	Image catalog from INPE - View and order to down...	0.1.4.1	Show quicklook image and make order to download
not installed	Spectral Profile	0.5.0	The Spectral Profile Plugin plots spectral profiles fro
not installed	Threading_demo	0.2	Simple demo to show a plugin operating in a separa
not installed	QuickWKT	1.0	Creates in-memory layers where you can paste you
installed	manageR	1.0	Interface to the R statistical programming language
installed	Google Layers	0.7.2	Add Google Layers into QGIS
installed	Test Plugin	0.3	This plugin has no real use yet
installed	Spatial Data Analysis for Point Pattern	0.196	SDA4PP with R functions. Requires R, RPy2 and R
installed	OpenStreetMap plugin	0.5	Viewer and editor for OpenStreetMap data
installed	MapServer Export	0.4.3	Export a saved QGIS project file to a MapServer ma
installed	Interactive Identify	1.3 (1.2)	Customizable descriptive information for features.
installed	Plugin Installer	1.2 (1.1)	Downloads and installs QGIS python plugins
installed	GdalTools	1.2.20 (1.2.17)	Integrate gdal tools into qgis

The plugins will be installed to ~/.qgis/python/plugins

Figure 5. 3. List of Plug-in in External Authors' Python Plug-in Repositories (Snapshot of Plug-ins Tab in Python Plug-in Installer in QGIS Version 1.7.0 in March 2011)

User contributed plug-in is first structured under one official repository of the project. After the wide spread development of Python plug-in, there are several QGIS plug-in repositories in 2008. Repository requires a storage area in the web server and one definition file written in the structure of XML. QGIS Python Plug-in Repositories are shown in Figure 5.4. The plug-in that is available in these repositories can be installed with the Python plug-in Installer.



Python Plugin Repositories

On this page you find a probably not complete list of available python plugin repositories. Some of them might already be listed in the plugin installer by default.

QGIS Official and Contributed Repositories

- QGIS Official Repository http://pyqgis.org/repo/official
- QGIS Contributed Repository http://pyqgis.org/repo/contributed

External Author Repositories

If you developed a python plugin for QGIS and want to make it available for other users, please add a link to your repository to the list below. The QGIS User Guide tells you how to integrate new repositories into the QGIS Plugin Installer.

- Faunalia Repository http://www.faunalia.it/qgis/plugins.xml
- Carson Farmer's Repository http://www.ftools.ca/cfarmerQgisRepo.xml
- Aaron Racicot's Repository http://qgisplugins.z-pulley.com/
- Barry http://www.maths.lancs.ac.uk/~rowlings/Qgis/Plugins/plugins.xml
- Martin http://mapserver.sk/~wonder/qgis/plugins-sandbox.xml
- GIS-Lab.info http://gis-lab.info/programs/qgis/qgis-repo.xml
- Volkan Kepoglu's Repository http://ggit.metu.edu.tr/~volkan/plugins.xml
- Bob Bruce's Repository http://www.mappinggeek.ca/QGISPythonPlugins/Bobs-QGIS-plugins.xml
- Sourcepole repository http://build.sourcepole.ch/qgis/plugins.xml
- Dimitris Kavroudakis repository http://www.dimitrisk.gr/qgis/plugins.xml
- Dane Springmeyer - http://qgis.dbsgeo.com/
- gis-plugins.nl - http://www.gis-plugins.nl/pyqgis

Figure 5. 4. List of External Author Repositories in August 2010 (Quantum GIS 2010a)

The development of SDA4PP plug-in is explained according to one implemented SDA tools, named as Kernel Smoothed Density. When this tool is explained as an example to demonstrate the structure of plug-in, the rest of implemented tools can also be understood as the same manner in the structure of the plug-in. The list of required files shown below is the minimum requirements for the developments of Python plug-in in QGIS. This simplified version is listed in order to explain the development of SDA4PP plug-in easily. Four files, namely density.py, densityRpart.py, density.ui and ui_density.py, are required for one of implemented SDA tools which is Kernel Smoothed Density Tool. Rest of listed files is necessary for any Python plug-in. The following files are needed at minimum in order to have a Python plug-in in QGIS:

- Initial Python Module; __init__.py (making the plug-in recognizable)

- Resource File; resources.qrc (defining icons), resources.py (resource file converted to Python syntax) and icon_1.png (can be several icons and each icon is used for one tools)
- Starting Main Python Module; sda4pp.py (creating menu window in QGIS, listing tools in window and setting up event driven actions in the menu)
- One of implemented SDA tools: Kernel Smoothed Density Tool; density.py (for GIS part), densityRpart.py (for SDA part), density.ui (for GUI with XML code) and ui_density.py (for GUI with Python code)

QGIS scans certain directories for finding C++ and Python plug-in. Each Python plug-in should be placed under a subdirectory of QGIS Python plug-in directory (default value is ~/.qgis/python/plugins, but, it may change according to used installer and operating system) and has its own directory. While QGIS starts, it scans the directory and finds the plug-in by reading __init__.py Python file. This file makes QGIS to recognize the plug-in in the QGIS plug-in manager. The user can activate the listed plug-in in the plug-in manager to use them in the QGIS. Figure 5.5 displays the source code of Python file, __init__.py that is developed for the SDA4PP plug-in. Name, description, version, required minimum version of QGIS to work for the plug-in, author name, icon and homepage of the plug-in should be defined in the __init__.py (Line 1 to 15) in order to make a connection with the plug-in manager. Other requirements are to add “classFactory” method (Line 16), import the main Python file, sda4pp.py, where the real work is carried out (Line 17) and sent “iface” object to “SDA4PP” class as reference object (Line 18). The “iface” object is the starting reference object for Python plug-in in order to start to communicate with the QGIS interface.

```
def name ():
  return "Spatial Data Analysis for Point Pattern"
def description ():
  return "SDA4PP with R functions. Requires R, RPy2 and
          R packages; Rgdal, Maptools, Spatstat"
def version ():
  return "0.196"
def qgisMinimumVersion ():
  return "1.0"
def authorName ():
  return "Volkan Osman Kepoglu"
def icon ():
  import resources
  return ":/icons/mainMenu.png"
def homepage ():
  return "http://ggit.metu.edu.tr/~volkan"
def classFactory (iface):
  from sda4pp import SDA4PP
  return SDA4PP (iface)
```

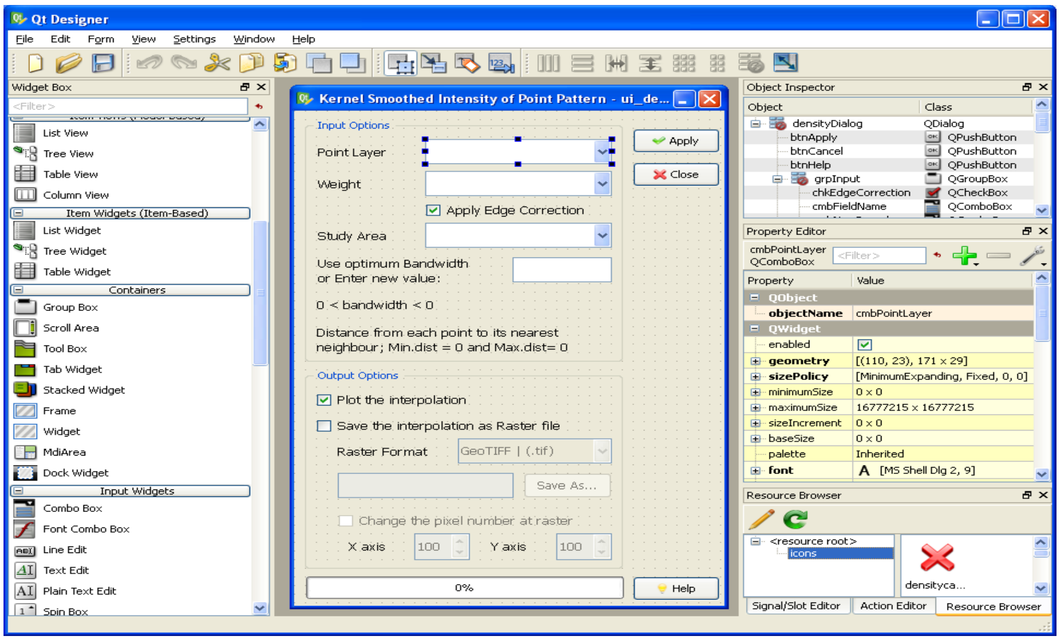
Figure 5. 5. Source Code of Python File; __init__.py

Second required file in the Python plug-in is the resource file. This file is needed in order to define the icon for each SDA tool. It uses a prefix as path definition in order to prevent name mixtures with other plug-in. For example, several icons are defined in the resource file as shown in the left side of Figure 5.6. 22 x 22 pixel sized PNG image format is convenient for icons, but other image formats can also be used. After defining the resource file, it should be converted to Python syntax. PyQt resource compiler makes this conversion with the following commands; “pyrcc4 –o resources.py resources.qrc”, and the result of conversion as the source code of resources.py is shown in the right side of Figure 5.6.

`<RCC>`
`        <qresource prefix="/icons" >`
`                <file>icon_1.png</file>`
`                ...`
`                <file>icon_n.png</file>`
`        </qresource>`
`</RCC>`
	`from PyQt4 import QtCore`
	
	`qt_resource_data = "\`
	`\x00\x00\x04\x7d\`
	`\x89\`
	`\x50\x4e\x47\x0d\x0a\x1a\x0a\x00\x00\x00\`
	`x0d\x49\x48\x44\x52\x00\`
	`\x00\x00\x20\x00\x00\x00\x20\x08\x02\x00\`
	`x00\x00\xfc\x18\xed\xa3\`
	`...`

Figure 5. 6. Source Code of Resource File (Left Side) and Result of Conversion to Python Syntax (Right Side)

Qt Designer, which is the same tool that C++ developer uses and comes with the installation of Qt Framework, is used to create the GUI for each tool. It is a visual design tool and allows dragging and dropping predefined widgets on dialog box and defining their properties. The GUI of Kernel Smoothed Density tool is shown as an example in Qt Designer in Figure 5.7. Text labels, text edit controls, combo boxes and push buttons as widgets are placed, sized and adjusted on the dialog box.



Object	Class
densityDialog	QDialog
btnApply	QPushButton
btnCancel	QPushButton
btnHelp	QPushButton
grpInput	QGroupBox
chkEdgeCorrection	QCheckBox
cmbFieldName	QComboBox

cmbPointLayer
QComboBox

Property	Value
QObject	
objectName	cmbPointLayer
QWidget	
enabled	✓
geometry	[(110, 23), 171 x 29]
sizePolicy	[MinimumExpanding, Fixed, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [MS Shell Dlg 2, 9]

Figure 5. 7. GUI of Kernel Smoothed Density in Qt Designer

All settings of widgets are stored by Qt Designer as XML based GUI file. This file (density.ui) is converted to Python (ui_density.py) with pyuic4 utility of PyQt with this command; "pyuic4 –o ui_density.py density.ui". Several code lines of these two files are shown below in Figure 5.8 in order to display as examples for the structure of GUI of Qt and PyQt. Left side of the figure shows XML based source code of GUI file of Qt and right side is for Python source code of GUI file of PyQt. Python GUI files can also be developed by writing the code without using Qt Designer.

```
<? Xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <author>Volkan Kepoglu</author>
 <class>densityDialog</class>
 <widget class="QDialog"
name="densityDialog">
  <property name="geometry">
   <rect>
    <width>394</width>
    <height>574</height>
   </rect>
  </property> ...
```

```
# -*- coding: utf-8 -*-

from PyQt4 import QtCore, QtGui

class Ui_densityDialog(object):
    def setupUi(self, densityDialog):
        densityDialog.setObjectName
                ("densityDialog")
        densityDialog.resize(394, 574)
        ...
```

Figure 5. 8. Source Code of GUI for Qt (Left Side) and Python (Right Side)

Figure 5.9 is simplified version for source code of starting main Python module of plug-in; sda4pp.py. The source code of module is much longer that shown in this figure. Complete source code of main Python module for GIS part of the plug-in is shown in Appendix A. The aim of simplifying the source code in the figure is to demonstrate the most important part of the module by not repeating the similar declarations. Python programming language is structured according to the indentation. The language is case sensitive and comment line starts with "#" character. The main function of this module is to create a new menu in the main window of QGIS and make a connection for user to access to each SDA tool. This module imports PyQt4 interface (Line 3, 4), QGIS interface (Line 5), resource file (Line 7), each module for implemented SDA tool separately (Line 9) and several modules found in Python standard library (Line 11). As explained previously, resource file is needed to access the icons. Each SDA tool is implemented in separate Python module. Therefore, each of them should be imported to sda4pp in order to execute the tool by linking from the menu. Two Python modules are imported to take additional functionality; one of them is “os” module which is the miscellaneous operating system interfaces. This module is mainly used for reading and writing a file, and manipulating paths. Other one is “time” module which is used for time related functions.

In Figure 5.9, only one of the implemented tools (density) is imported as an example (Line 9). Reference object of QGIS interface (iface) is sent as argument to “SDA4PP” class of this module in Line 13 and saved within variable in Line 15. In similar way, QgsMapCanvas class which is the main class for displaying all GIS data types is referenced and saved in the first function of class of this module (Line 16). Following “initGui” function initializes the settings related with GUI of new menu for the plug-in (Line 17). Line 19 creates an action for starting of density tool. In this line, QAction class takes three arguments; the icon for the tool shown in the menu, text that is shown in the menu and a reference to the parent main window of QGIS. Line 22 is the signal and slot mechanism of Qt Framework. When user clicks to “Kernel Density” text in the menu, triggered signal is emitted in “densitytool” action and slot part responds this signal to execute new function named as “doDensityTool” declared in Line 31. These declarations (action and signal/slot definitions) should be repeated for each tool (Line 20 and 23). A new menu named as “SDA4PP” is created in Line 25. Each action as a list in the order is added to new created menu in Line 26. The menuBar of QGIS is referenced and “SDA4PP” menu is added before the last menu of QGIS (Line 27 – 30). When “doDensityTool” function is called, “DensityTool” class in the density module is initialized (Line 32) and executed (Line 33). The “iface” object should also be sent as an argument to “DensityTool” class, because density dialog (GUI part of the tool) needs to make a connection with the QGIS interface. This is the brief explanation of initializing user event handling for each tool and creating a new menu as written in the starting main Python module of plug-in; sda4pp.py.

```
# -*- coding: utf-8 -*-
# imports interface of PyQt and QGIS
from PyQt4.QtCore import *
from PyQt4.QtGui import *
from qgis.core import *
# initialize Qt resources from resource file
import resource
# import each developed SDA tool: e.g., density tool
import density...
# python modules from python standard library
import os, time
class SDA4PP:
  def __init__ (self, iface):
    # save reference to the QGIS interface
    self.iface = iface
    self.canvas = iface.mapCanvas ()
  def initGui (self):
    # create action for starting of density tool
    self.densitytool = QAction (QIcon (":/icons/menuDensity.png"),
                                  "Kernel Density", self.iface.mainWindow ())
    ...
    # event part: Signal and Slot mechanism
    QObject.connect (self.densitytool, SIGNAL ("triggered ()"),
                              self.doDensityTool)
    ...
    # setup a new menu in the main window of QGIS
    self.menu = QMenu ("SDA4PP")
    self.menu.addActions ([self.densitytool ...])
    menuBar = self.iface.mainWindow ().menuBar ()
    actions = menuBar.actions ()
    lastAction = actions [len (actions) - 1]
    menuBar.insertMenu (lastAction, self.menu)
  def doDensityTool (self):
    d = density.DensityTool (self.iface)
    d.exec_ ()
  ...
```

Figure 5. 9. Simplified Version of Code for Starting Main Python Module of Plug-in

Last listed files are required for Kernel Density tool. Each tool is composed of two Python modules; one file is used for GIS tasks and other one is for SDA tasks. The simplified version for the source code of GIS part of Density Tool is shown in Figure 5.10. This module imports the GUI part of the tool in Line 2, the SDA part of the tool in Line 4, and main Python module of plug-in which organizes all functions related with the QGIS interface for each tool (sda4ppReadQGISLayer) in Line 6. The class of this module is declared in Line 7 with two GUI arguments. GUI interface is initiated in Line 9 and iface object is saved as argument of the class in Line 11. Line 13 and 14 setups the user interface settings. The size of the dialog cannot be changed due to declaration in Line 14. Many variables are initialized with their default values and stored in one Python dictionary typed variable (self.index) in Line 16.

```
1     # GUI part: import the code for the dialog
2     from ui_density import Ui_densityDialog
3     # Run R code
4     import densityRpart
5     # Read qgis layer
6     import sda4ppReadQGISLayer
7     class DensityTool (QtGui.QDialog, Ui_densityDialog):
8       def __init__ (self, iface):
9         QtGui.QDialog.__init__ (self)
10        # save reference to the QGIS interface
11        self.iface = iface
12        # set up the user interface
13        self.setupUi (self)
14        self.setFixedSize (self.size ())
15        # Init variables
16        self.index = {"inputFilePath":"","inputFileName":"", "projText":"",
                    "rstExtension":"tif", "plot": True, “save”: False ...}
17        # Read the properties of qgis layer before displaying the dialog
18        self.readInputFile ()
19        # Event part
20        QtCore.QObject.connect (self.btnApply, QtCore.SIGNAL ("clicked ()"),
                                   self.runApply)
21        # Set signal and slot mechanism for other GUI widgets
```

Figure 5. 10. Simplified Version of Code for GIS Part of Density Tool

```
    ...
def readInputFile (self):
    # Init rpy2
    import rpy2.robjects as robjects
    self.R = robjects.r
    # Get point layer
    self.ReadPointLayer = sda4ppReadQGISLayer.ReadQgisLayer
                                                          (self.iface)
    layers = self.ReadPointLayer.getLayerList
                                       (CheckNumericField="YES")
    if layers == []:
        QtGui.QMessageBox.information (self.iface.mainWindow (),
                                       "SDA4PP Plug-in Error",
                                       "Please add point typed vector layers.")
        return
    for layer in layers:
        self.cmbPointLayer.addItem (layer)
    # get other properties of layer required for this type of analysis
    ...
    # Projection
    self.index ["projText"] = self.ReadPointLayer.getProj ()
def runApply (self):
    # InputFile
    if self.cmbPointLayer.currentIndex () == -1:
        QtGui.QMessageBox.information (self.iface.mainWindow (),
                               "SDA4PP Plug-in Error",
                               "Please add point typed vector layer in QGIS.")
        return
    # check other controls related with dialog options
    ...
    # Apply analysis part with R
    densityRpart.doSpatialAnalysis (self.R, index= self.index)
    # Save
    if self.index ["save"]:
        resSave = self.ReadPointLayer.saveRaster
                               (self.index ["outputFilePath"])
```

Figure 5. 10. Simplified Version of Code for GIS Part of Density Tool (continued)

Before setting signal and slot mechanism for each widget in Line 20, the properties of QGIS layers should be read by referencing to the main GIS part of Python module; sda4ppReadQGISLayer. Therefore, a new function named as readInputFile is called in Line 18. First two lines of this function imports RPy2 module (Line 25) and saves the RPy2 objects as reference object (Line26). RPy2 is the Python bindings for R interface. The “ReadQGISLayer” class is initialized in Line 28. By using this class, the properties of vector map layer are taken with calling the related functions of the module. For example, this class gets the name of layer as list variable by calling “getLayerList” function of module in Line 29. In the next line, “if” statement checks whether any layer name is returned or not. If there is no point typed vector map layer in the QGIS, the tool gives an informative message about the situation (Line 31) and the execution of the tool is stopped (Line 32). If there is the layer in the QGIS, their names are added to combo box widget by iterating each name (Line 33 and 34). Other properties of layers required for the analysis are taken in a similar way; calling the related function in sda4ppReadQGISLayer module. For example, the coordinate reference system of first added layer to combo box is taken and saved as variable in the Python dictionary variable (Line 38). When all related properties are called, some of the properties are assigned to load in the widgets and some of them are stored as variable for further use in the analysis. This function ends when all properties are taken. The execution of module returns to Line 20, and continues from here.

Line 20 is the place where the declaration of event part has started. QObject.connect method takes three arguments; the name of widgets, type of signal specific to widget and the name of function. This function is executed when the signal is triggered by the widget. In this example, when “Apply” push button is clicked by the user, runApply function would be executed. This function is declared in Line 39. There are many control sections in this function which checks all variables related with the dialog options. For example, if statement in Line 41 checks that whether there is a point layer in the combo box widget or not. If there is no point typed vector layer in QGIS, an error message is displayed to the user and the execution of the tool is stopped. Otherwise, the flow continues to check other controls related with dialog options.

At the final stage, index variable and reference of RPy2 module object are sent as arguments to start to process the analysis of the SDA part of the tool (Line 49). If the “plot” option is kept in its state (by default it is checked), the output of analysis is displayed as graphics by using native R graphics device. The graphics can be saved as image file. The displayed graphic can be either a chart or view of spatial data. Another common option in the dialog of tool is the “save” option. If this option is checked, the output of analysis is converted to GIS data format in the SDA part of tool and asked to user to add as a map layer in the QGIS or not. If accepted by the user the newly created GIS data format either ESRI Shapefile or georeferenced TIFF image file is added as map layer in the table of content and the output is displayed in the map render of QGIS (Line 51 and 52).

Figure 5.11 is the simplified version for source code of SDA part of Kernel Density Tool. In Line 2, sda4ppRoptions module is imported. This is the main SDA part of Python module for organizing common functions related with R interface like converting GIS data format to spatial object, plotting the graphics, setting the size of R native graphic device and closing the R device. Line 4 imports another plug-in module named as “wl” which is developed to get the results of any object. This module returns the value of variable or object as a string and writes this value in a text file. It is developed to help the programmer for logging purposes. Similarly, the result of returned value of any object can be seen in the terminal by using native “print” command of Python in Linux OS, whereas, the same mechanism is not working during the execution of the tool in the QGIS environment for Windows OS. For this reason, the “wl” module is developed to get the values of the variables or object in the Windows OS.

```
1       # R options
2       import sda4ppRoptions
3       # write to logFile.log
4       import wl
5       def doSpatialAnalysis (r, index):
6         wl.wl ("index: " +str (index), dt)
7         # Projection
8         r ("p4s <- CRS (\"" + str (index ["projText"]) + "\")")
9         r ("rect <- as (v_spdf [\"" + str (index ["fieldName"]) + "\"], \"ppp\")")
10        ...
11        # Plot
12        if index ["plot"]:
13          rcodes = ["plot (z, main=\"Kernel Density for” \
                      + str (index ["inputFileName"]) + "\")"]
14          sda4ppRoptions.plotGraphics (r, rcodes)
15        # Save
16        if index ["save"]:
17          r ("sgdf <- as (z, \"SpatialGridDataFrame\")")
18          r ("proj4string (sgdf) = CRS (\"" + str (index ["projText"]) + "\")")
19          r ("writeGDAL (sgdf, \"" + str (index ["outputFilePath"]) + "\",
                 drivername = \"" + str (index ["rstfile"]) + "\", type = \"Float32\")")
```

Figure 5. 11. Simplified Version for Source Code of SDA Part of Density Tool

R reference object and Python dictionary variable are taken as arguments in the function of SDA part of tool (Line 5). When R code is executed by calling R objects in the related packages, R code is closed in parenthesis with “r” object of RPy2 module. For example, the projection of map layer is sent to R environment in Line 8. The path of GIS vector layer is sent to R to read into a suitable spatial vector object by using “readOGR” function in the “sda4ppRoptions” module. The module calls the “readOGR” function of rgdal package. By this way, spatial object in the format of vector spatial point data frame named as “v_spdf” can be directly called in the SDA part of tool. This object is converted to native format of spatstat package (Line 9).

The execution of the module continues to process the analysis until reaching the end of module where the common dialog options like “plot” and “save” options are found. The “plot” option is controlled with “if” statement in Line 12, whether this option is checked or not. Plotting the analysis result of density tool is declared as R code and stored in the Python list variable (Line 13). This code is sent to the “plotGraphics” function of “sda4ppRoptions” module for execution of the code (Line 14). Similarly, the execution of “save” option is controlled with “if” statement in Line 16. If “save” option is checked in the dialog of tool, the analysis result is saved as GIS data format. The analysis result of density tool is real-valued pixel image, which is the raster layer of spatstat package. At first, pixel image format is converted to spatial raster object (Line 17), its projection is defined (Line 18), and then, spatial raster object as grid data frame is converted to raster GIS data format either georeferenced tagged image file or Erdas Imagine file according to selected options in the GUI of the tool (Line 19). The pixel number of X and Y axis at the raster layer can also be changed. Default pixel value is 100. At the final stage, the analysis result of the tool is added as map layer in the main window of QGIS. By this way, the execution of density tool is completed by showing the result of analysis as raster layer in the QGIS.

In general, all needed modules are imported at the beginning section of the module, before the declaration of class of module. After class declaration, referenced objects are saved, GUI part of tool is initialized, and the used variables with their default values are declared. Initial properties of map layer are read and saved as variable before displaying the GUI. Then, event part is declared with signal and slot mechanism of Qt framework. At this stage, the class of tool’s declaration is finalized and the GUI of tool is displayed. According to the choices of users in the dialog, events are emitted as signals and related slots are executed. The result of interaction between the widgets and the user is saved as variable under one Python dictionary variable. Dictionary variable storing all options are sent to SDA part of tool for execution of the analysis. Analysis part starts with converting GIS data format to suitable spatial objects. Analysis options are sent to R environment as parameters for the main execution of analysis function. After the analysis is executed, the result is displayed as output in the form of drawing charts, displaying statistical figures and maps or converting objects back to GIS data format. Each analysis tool has specific options, properties and related widgets in their dialog. Repeating common properties and options are organized under specific module. Common functions for GIS related tasks can be found in Python module named as sda4ppReadQGISLayer and for SDA related tasks are in sda4ppRoptions module.

5.2. Architecture of SDA4PP Plug-in

QGIS, PyQGIS, PyQt and Qt Framework are required for the development of plug-in on the GIS part. R, R packages and RPy2 are required for SDA part of the plug-in. This seperation has also shaped the inner design schema of the plug-in as shown in Figure 5.12. Each developed tool in the plug-in is composed of two Python modules; GIS and SDA module. This makes simple to find where the error comes from. The result of all GUI actions performed by the user are stored as variables in the GIS part. All options of each tool are gathered under one Python dictionary variable and sent to SDA part to proceed as parameters. GIS part of each tool can interface seperately with QGIS, but, this will cause to repeat the similar functions many times for each tool. Instead of distributing the same functions to many files, one Python module is developed which organizes all QGIS interface functions under one module. Each tool links to this module and interfaces to QGIS through this module. By means of developing this one Python module, sda4ppReadQGISLayer, a new tool can be systemmatically added to the plug-in and the structure of the plug-in has become more stable for the changes in the interface of QGIS.

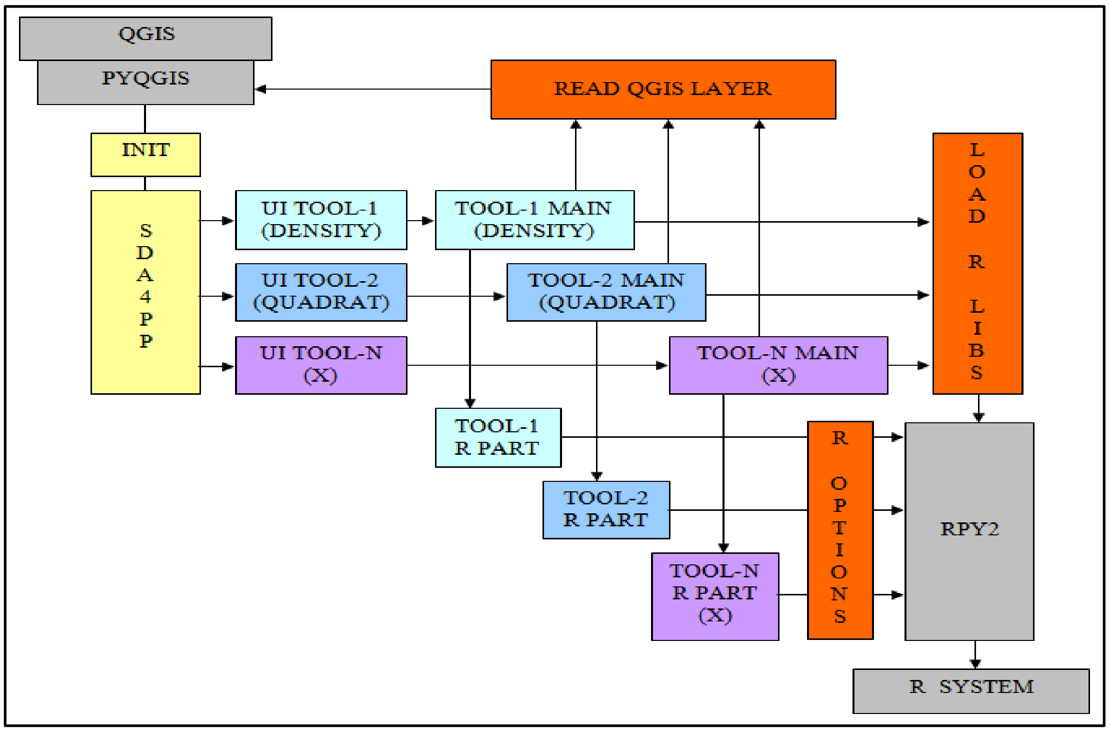


Figure 5. 12. Inner Design Schema of SDA4PP Plug-in

Accessing to QGIS API with Python can be possible with iface object. At first, iface object is used in the Python module named as __init__.py, which is the initial Python module of the plug-in that initiates to communicate with QGIS API. This module passes the iface object to starting main Python module named as sda4pp, which creates a new menu window in the QGIS and provides the signals and slots mechanism of Qt Framework to interact with the user. High level event handling mechanism in the Qt framework is called as signals and slots (Summerfield 2008). The term “signal” is defined by Qt as that specific signals are attached to related Qt’s widgets and they are emitted when a particular event occurs. The term “slot” is defined as a function that is called in response to a particular signal (Qt 2011). In Qt, slots are methods that must be declared with a special syntax. But, in PyQt, slots can be callable like any function or method, and no special syntax is required (Summerfield 2008).

In order to communicate with required interfaces, the GIS part of tool imports the core and gui classes of PyQt4 and qgis, whereas the SDA part of tool imports only RPy2 Python module. QgsMapCanvas is the main class for displaying all GIS data types. This class is accessed through iface object. In general, SDA tools developed in the plug-in starts at first executing the display of GUI of tool, and then, selecting point layer from the combo box widgets in the GUI, selecting other options, and finally applying to perform to analysis and the result is shown as a new map layer in the QGIS or plotting a graphic with some descriptive statistics. The selected map layer in the table of content of QGIS can be found by looping the layers in QgsMapCanvas class. QgsMapLayer is base class for all map layer types. The map layer in the loop can be catched according to the layer types like vector and raster layer. Vector layers can be filtered according to the shape of feature geometry like point, line and polygon typed vector map layer.

Each map layer has the following properties such as extent, coordinate reference system, source, etc. Extent of layer defines all features entire rectangular geographic extent with reference to minimum and maximum coordinates of the edges of rectangular. Coordinate system of layer is a mathematical formula that transforms feature locations between the earth and map flat surface. This transformation is handled by means of using PROJ.4 library. Coordinate system and extent of layer can be obtained respectively with “srs” and “extent” functions of QgsMapLayer class. Another property of the layer is the source of layer, which means that each map layer has also file name and path that can be derived from the "source" function of QgsMapLayer class. Other common property for vector layer is to have an attribute table. Attribute table of vector layer is provided by QgsDataProvider which is another abstract base class for spatial data provider implementations. It has two types; vector and raster data provider. Each vector layer has attribute table provided by vector data provider class. Each attribute table has field name and type like integer, real and string. These are the most used common properties of each map layer of QGIS.

The sda4ppReadQGISLayer Python module interfaces with these properties of the layer through mentioned classes. This has prevented to develop same functions many times for each tool. During development of plug-in in 2.5 years, QGIS interface has changed several times; especially before reaching most mature stable version 1.0. Any change in the implemented part of QGIS interface should also be adopted for each tool. Forming sda4ppReadQGISLayer module as one module handling all connections with QGIS for all tools has avoided to correct many functions for each tool when any changes has occurred in QGIS interface. With this structure, the plug-in has become more stable to any break down of QGIS interface changes. The sda4ppReadQGISLayer Python module interfaces with QGIS for all tools in order to initiate the following functions for each developed tool; getting layer names as Pyhon list variables, filtering the selection of layer according to the feature shape of geometry and the name of layer, having the source of layer, extent of layer, projection of layer, counting the number of features in the layer, getting field names and types in the attribute table, returning raster layer names and sources, selecting features according to unique identifier of feature, saving the vector map layer as ESRI Shapefile, zooming to the extent of layer, adding the map layer to the table of content of QGIS, and saving the raster map layer as either georeferenced tagged image or Erdas imagine file format.

SDA part of each tool is executed with R thorugh RPy2 Python module. Similarly, common tasks related with SDA part of the plug-in are organized under two modules that prevented to interface to R seperately for each tool. Converting GIS data format to spatial data frame format of R and plotting the graphics as a result of the analysis are some of the common tasks of SDA part that are handled with a separate Python module named as sda4ppRoptions. Another developed Python module for SDA part is named as sda4ppLoadRlib whose function is to load the specific R packages required for the related tool before executing the analysis. Every tool does not need to load all packages. Rgdal, maptools and spatstat packages are mostly required packages for many tools. Other packages like rggobi and iplots are used for a few tools. Rgdal package handles to input and output of spatial objects. Maptools is required to convert from spatial object of S4 classes to an object of class “ppp” format for point pattern of spacestat package or vice versa. Spatstat package is used to perform the execution of the analysis part of the tool.

In the SDA part of the plug-in, GIS data format is first converted to spatial data frame objects by means of rgdal package, and then to point pattern format by means of maptools package. The SDA analysis is done at the point pattern format by using spacestat package. The output is produced in the R and sent to GIS part of the tool for displaying the result in the QGIS. The list of input and output formats for each SDA tool is summarized in Table 5.1. The output of analysis can be either producing a new GIS data or plotting a graphics and maybe with some explorative statistical numbers and texts. Graphics are drawn by using native R plot graphic device. Statistical explanations are extracted from the point pattern format. It is processed either converting as string for displaying in the GUI of executed tool or attaching as attribute table of vector layer. Derived new GIS data is converted either from point pattern format to ESRI Shapefile as vector layer or from spatial grid data frame to raster layer by means of using rgdal package. Raster layer can be in the format of either georeferenced TIFF image file or Erdas Imagine Images. This produced GIS data is added in the table of content of QGIS, zoomed to its extent and displayed as a new map layer of QGIS as a result of the performed analysis.

Table 5. 1. List of Input and Output Formats for each Developed SDA Tool

Developed SDA Tools	Input				Output			
	Point Layer	Polygon Layer	Raster Layer	User Input	Raster File	SHP File	CSV File	Stat.
Uniform Intensity	x							x
Kernel Smoothed Density	x	x			x			
Adaptive Density	x				x			
G & F Estimate Function	x							
Ripley's K & L Function	x							
Local Ripley's K and L	x							
Quadrat Test	x					x		x
Kolmogorov-Smirnov Test	x							x
Sim. Envelope of CSR	x							
Fit Poisson Model	x				x			x
Kriging	x		x		x			
Generate Random Point				x		x		
Generate Regular Point				x		x		
Nearest Point & Distance	x						x	x
Polygon to Point		x				x		
Point to Polygon	x					x		
Rggobi	x						x	
Linked Stat. Distance	x							
R Console				x				x

Some spatial packages like spatstat and splancs data types belong to old style S3 classes. Bivand et. al (2008) explain that first version of S language did not include the use of class/method mechanisms and adds that S3 classes are implementation of S language version 3 that is known as old-style classes. Disadvantage of S3 class is that this class has no formal description and their methods could not recognize the inheritance (Chambers 2008). That’s why, maptools package is required for conversion from S4 classes of rgdal package to ppp format of S3 class of spacestat package. Spatial objects based on S4 classes are regulated under sp package which rgdal and maptools use this structure. It is trying to organize all spatial related S3 objects under the structure of S4 classes of sp package. For example, vector map storing fire locations as point data layer in the format of ESRI Shapefile, is read with “readOGR” function of rgdal package and the output is spatial point data frame (SPDF). Attribute table of vector map layer is attached as spatial data frame in the data slot of SPDF. Bivand et. al (2008) define spatial data frame as designed to behave like data frame. Chambers (2008) defines data frame as a two-way array with columns corresponding to variables. Data frame have attributes like rows and columns. SPDF should be converted to spatial point object and then to the ppp format. The structure of SPDF and ppp format is shown in Table 5.2 and 5.3 respectively. Data slot of SPDF keeps the attribute table of layer, coords slot stores the coordinates of each feature location, bbox is the extent of layer, proj4string slot is the reference of coordinate system, whereas in point pattern format, the window is the extent of layer, the list named as “n” is the number of feature, x and y lists have the coordinates of each feature location.

Table 5. 2. The Structure of Spatial Point Data Frame of S4 Class

```
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 30 obs. of  5 variables:
  .. ..$ MAHALLE: Factor w/ 4 levels "A","C","D","E": 4 4 4 1 1 4 1 1 3 4 ...
  .. ..$ IDS    : int [1:30] 8000000 40 53 58 70 92 93 101 106 109 ...
  .. ..$ POINT_X: num [1:30] 487088 487026 487054 487573 487840 ...
  .. ..$ POINT_Y: num [1:30] 4417100 4416718 4417443 4416843 4418123 ...
  .. ..$ VALUE  : int [1:30] 80000000 90000000 95780000 87409000 89120 22571
74997000 92375100 33250000 112604 ...
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:30, 1:2] 487088 487026 487054 487573 487840 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
  ..@ bbox       : num [1:2, 1:2] 486983 4416614 488042 4418769
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
  .. .. ..@ projargs: chr " +proj=utm +zone=36 +ellps=intl +units=m +no_defs"
```

In conclusion, the conversion of spatial data from QGIS to R or vice versa is proceeded at the background by using specified processes. As mentioned, there is also a conversion in the structure of spatial objects in the R environment. This conversion like from S4 to S3 type classes and vice versa is handled by using specified R packages.

Table 5. 3. The Structure of “ppp” format of S3 Class

```
List of 5
 $ window    :List of 4
  ..$ type  : chr "rectangle"
  ..$ xrange: num [1:2] 486983 488042
  ..$ yrange: num [1:2] 4416614 4418769
  ..$ units :List of 3
  .. ..$ singular  : chr "unit"
  .. ..$ plural    : chr "units"
  .. ..$ multiplier: num 1
  .. ..- attr(*, "class")= chr "units"
  ..- attr(*, "class")= chr "owin"
 $ n         : int 30
 $ x         : num [1:30] 487088 487026 487054 487573 487840 ...
 $ y         : num [1:30] 4417100 4416718 4417443 4416843 4418123 ...
 $ markformat: chr "none"
 - attr(*, "class")= chr "ppp"
```

As a summary, the GUI of tool is prepared in the Qt Designer. XML based GUI format is converted to Python code by using PyQt utilities. The GIS part of the tool imports the GUI and required all other modules at the beginning section of the module. It setups event handling mechanism and GUI options. GIS part also reads the properties of map layer of QGIS through main Python module of plug-in, sda4ppReadQGISLayer. All stored options are sent as variables to SDA part of tool. Common SDA tasks are also developed under one Python module named as sda4ppRoptions. This module deals with the conversion between GIS format and spatial object. The analysis is executed according to coding R syntax. R code is sent by both SDA part of tool and sda4ppRoptions Python module to R environment through using RPy2 Python module. The result of the analysis is converted to GIS data and displayed in the QGIS environment. As a result, this architecture schema has created a systematic for adding a new analysis tool. Adding new tool is a matter of spending more effort and time. New analysis tools can be added in the same manner with the explained structure.

5.3. FLOS Development of SDA4PP Plug-in

The benefit of FLOSS development is not only made the software freely available, but more important than is a platform which provides public collaboration. FLOS developers believe that if they allow other developers to modify their code, the software will be free of error and more useful. In this thesis, during the development of SDA system by following FLOSS methodology, it is seen that the information provided by the members of GIS and SDA community is overwhelming. Amazing FLOS GIS resources are available online. Members of QGIS, Python and R communities are eager to help the users and especially to newcomers. The responsibility of helping to the requested user is taken as a very serious task by these communities. They care about the development of their product and know that anyone can criticize anything in any time in this environment, because all result of carried work can be accessible from the Internet.

This is a matter of prestige for the member of the project. The person who knows the best gains a respect from other members of the developer and his opinion about the discussed subject is more important than others. The management of this FLOSS project is purely democratic but the respect to the most experienced developer and traditional rules that are shaped from the past achievements during the development of the project are also kept in mind. These thoughts have strengthened the quality of many FLOSS products in every day.

During the development of SDA system in 2.5 years by using FLOSS products, it is realized that the most important meaning of FLOS for the members of the community is to help each other. By this way, it is assumed that more useful and successful product can be developed. FLOS is a new form of freedom movement for learning and spreading their knowledge between the members. The simplest way of being a member of this community is to leave a good comment. This helping mechanism can become stronger either to find a solution to asked problem, explain how to fix the error or in other worlds help them to develop the software on their environments. The meaning of helping to develop the software is not only writing a code but also making donations to the project or it can be translating the documents or any other things related with the development of the project. In return, learned knowledge should be shared in similar way with them by documenting the information. Documenting and sharing the information is necessary especially for the newcomers to adopt them as a member of the community.

There are many ways to spread and share their knowledge such as by using blogs, forums, e-mail lists, preparing manuals and tutorials, even publishing all IRC logs. Simplest and most active way of learning and sharing the knowledge in the FLOSS environment is to use e-mails. At first, project’s e-mail lists should be subscribed, and then, messages between members followed and e-mails should be searched before asking a question and finally an e-mail message should be posted. E-mail lists are the most important infrastructures of the FLOSS projects, because they are mostly used and have latest information about development. The member of the communities provides free technical support to their users by using this communication media. For example, users request new features and share their ideas with the developers. Developers discuss the bugs and decide the new direction of development also in these e-mails. Separate groups have specific aims to use these e-mails. For this reason, e-mail lists are divided according to the specific discussion subjects. For example, QGIS community has ten e-mail lists in February 2011; User, Fossgis-talk-liste, Developer, Community team, Translation, Education, Release Team, Project steering Committee, Commits and Bug Tracking. E-mail lists are open to everyone and store the history of related subject for the project.

Communities are using all forms of digital tools for disseminating and facilitating their products. In this environment, all content can be discussed publicly between members and also be searched online. Especially the developers are expecting from their users to offer helpful suggestions. These suggestions are sometimes to become guidelines for them to make further development and sometimes to earn more profit. But, this feedback mechanism could not work automatically, unless an infrastructure system is setup and provided for newcomers to be adopted as a member of the community. Infrastructure systems such as blogs, forums, wikis, e-mail lists, etc. are also setup for this purpose; to get feedback from the users and develop the project together.

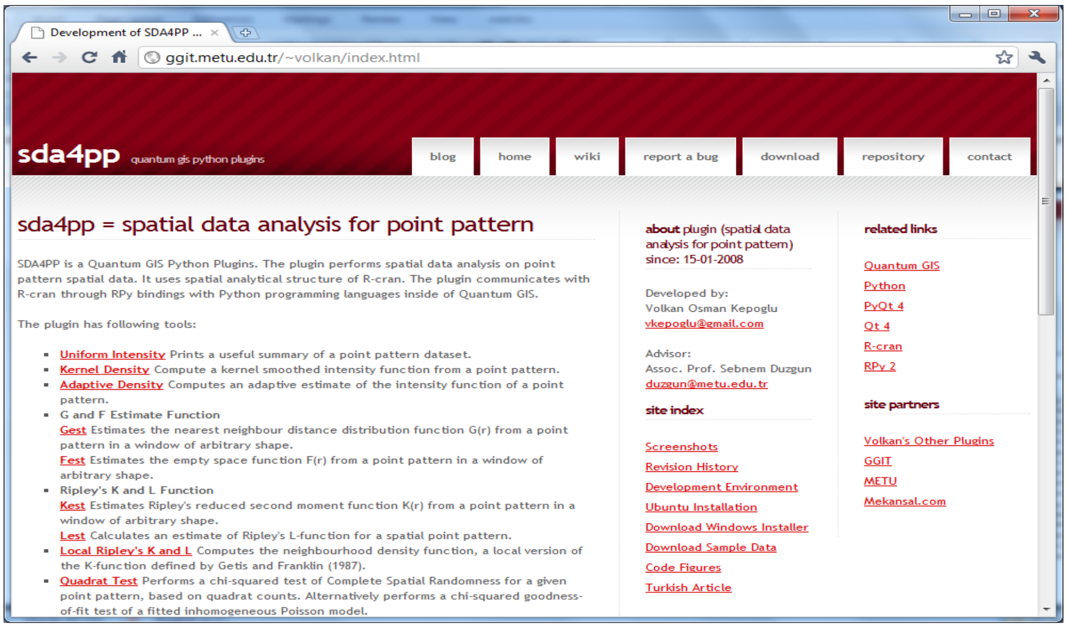
During the development of plug-in, FLOSS environment has helped the author of the thesis to learn how to make developments in the QGIS. New GIS software, customized version of QGIS, can be developed by using QGIS interface. This requires compiling the source code of QGIS. Another development option in the QGIS environment is to develop a new plug-in by using either Python or C++ programming language. The development of plug-in is preferred due to decided coupling strategy of the thesis. How to develop the plug-in and fix the compilation error is learnt from the members of the QGIS community. It is a self-learning experiment and when a dead end has reached, a help is requested from the community by using e-mail lists. The developers of the QGIS community explained clearly and in time how to solve the problems. Without interaction between the members in the e-mails, this development could not be achieved.

Applying FLOSS development methodology during the development of plug-in has resulted in several benefits. The users and members of the QGIS community have used, tested and reviewed the plug-in; especially, Paolo Cavallini and Giovanni Manghi have found several errors and reported via e-mails. Also, some of the errors are recorded as a bug entered by the users into the bug page of the plug-in web site. Several new features are requested by the users like the development of kriging tool and the support of using PostGIS layer during the analysis. Although Kriging tool belongs to Geostatistics field, it is developed in the plug-in, whereas the implementation of PostGIS layer could not be completed in time.

Two feedback mechanisms is setup for getting the information about the plug-in users. First mechanism is to collect information about the users like the name, e-mail address, job title, application area, organization and country of the users who filled the registration form in order to download the installer of SDA system. Second way is to record statistical figures by the Google Analytics about the visitors who viewed the web site of the developed plug-in. These figures about the usage of the plug-in are explained in detail in the next chapter.

The development of Python plug-in in QGIS has started by using FLOSS development methodology that is learnt while developing the plug-in by following the developments of QGIS, Python, PyQt, R and other FLOS GIS resources. The source code of SDA4PP plug-in can be accessible through Internet connection since June 2008. The implementation of the plug-in has started in January 2008. Six months later, the development is carried out according to the FLOSS development methodology. This method requires publishing the source code of the development, giving information about how the development is ongoing, and attracting the users and developers to help the development process by adapting each person as a member of the project. Source code is published under a dedicated repository which is prepared to realize this specific purpose. Information about the development is provided mainly with the developed web site. However, no message exists for the users and developers to develop the plug-in together. This is done deliberately in order to keep the originality of the thesis.

Dedicated web site is prepared under the domain of ggit.metu.edu.tr with the following web address; http://www.ggit.metu.edu.tr/~volkan/index.html. Snapshot of the main page of web site is shown in Figure 5.13. In this web site, developed tools are explained, screenshots of each tool are displayed, development environment with their versions of used components are listed, installation of SDA system for Ubuntu OS is explained, and an installer is distributed. The installer is prepared to install the SDA system in Windows OS. The installer is a setup file which re-packages all installer of each component under one executable file. It also installs all required R packages, defines “R_HOME” variable as system variable, and adds the path of R program to the path variable. In order to download the installer, a registration form should be filled by the user. This form is prepared to get the information about the user. This information is interpreted in detail in the next chapter.



Development of SDA4PP ...
ggit.metu.edu.tr/~volkan/index.html
sda4pp quantum gis python plugins
blog
home
wiki
report a bug
download
repository
contact

sda4pp = spatial data analysis for point pattern

SDA4PP is a Quantum GIS Python Plugins. The plugin performs spatial data analysis on point pattern spatial data. It uses spatial analytical structure of R-cran. The plugin communicates with R-cran through RPy bindings with Python programming languages inside of Quantum GIS.

The plugin has following tools:

- Uniform Intensity Prints a useful summary of a point pattern dataset.
- Kernel Density Compute a kernel smoothed intensity function from a point pattern.
- Adaptive Density Computes an adaptive estimate of the intensity function of a point pattern.
- G and F Estimate Function
  Gest Estimates the nearest neighbour distance distribution function G(r) from a point pattern in a window of arbitrary shape.
  Fest Estimates the empty space function F(r) from a point pattern in a window of arbitrary shape.
- Ripley's K and L Function
  Kest Estimates Ripley's reduced second moment function K(r) from a point pattern in a window of arbitrary shape.
  Lest Calculates an estimate of Ripley's L-function for a spatial point pattern.
- Local Ripley's K and L Computes the neighbourhood density function, a local version of the K-function defined by Getis and Franklin (1987).
- Quadrat Test Performs a chi-squared test of Complete Spatial Randomness for a given point pattern, based on quadrat counts. Alternatively performs a chi-squared goodness-of-fit test of a fitted inhomogeneous Poisson model.

about plugin (spatial data analysis for point pattern)
since: 15-01-2008

Developed by:
Volkan Osman Kepoglu
vkepoglu@gmail.com

Advisor:
Assoc. Prof. Sebnem Duzgun
duzgun@metu.edu.tr

site index

Screenshots
Revision History
Development Environment
Ubuntu Installation
Download Windows Installer
Download Sample Data
Code Figures
Turkish Article

related links

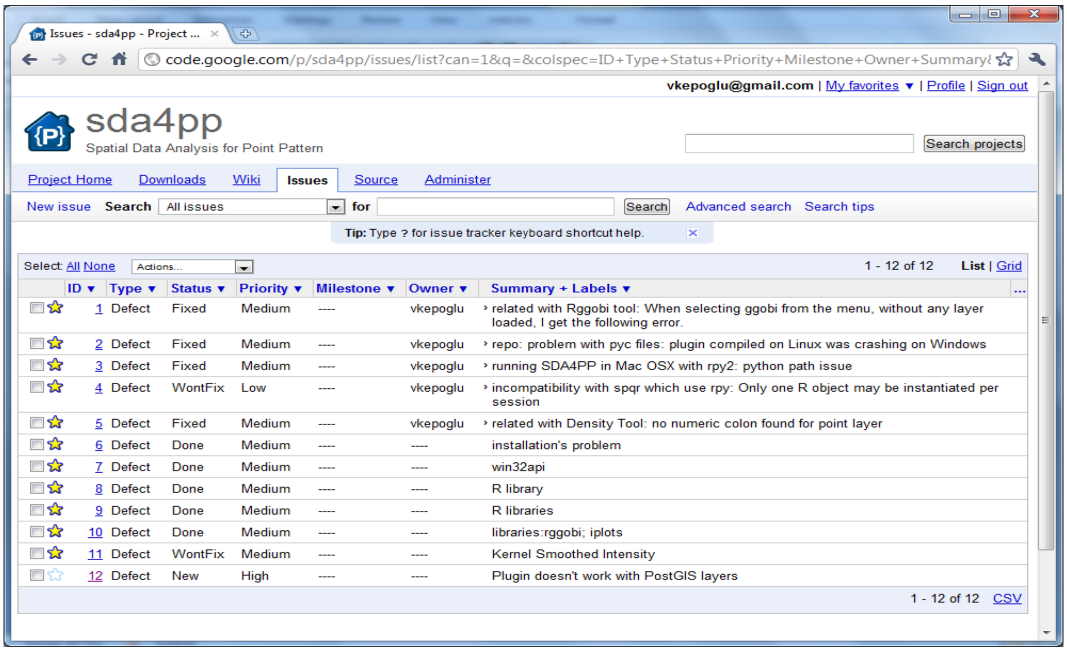
Quantum GIS
Python
PyQt 4
Qt 4
R-cran
RPy 2

site partners

Volkan's Other Plugins
GGIT
METU
Mekansal.com

Figure 5. 13. Main Page of Web Site Developed for FLOS SDA System

Web site has links to the blog, wiki, reporting a bug, revision history of plug-in, and the repository. FLOS tools are used for setting up these facilities. For example, blog is the free version of Wordpress. News about the development and new revisions of installer are announced in the blog. Wiki and reporting a bug pages are integrated to web site by using the structure of code.google.com. Bug reporting is a readymade form provided by the code.google.com to facilitate FLOSS projects. Several users have reported bugs to the author of this thesis by filling the form as shown in Figure 5.14. In the form, description, status and effects of the bugs can be setup by the user. When the bug is fixed, the status of the bug is changed as “fixed” to end the reporting. There are 12 bugs as of February 2011. For example, first error reported by Paolo Cavallini in June 11, 2009 via e-mail in the qgis-developer e-mail list is entered as a bug by the author of this thesis. The bug occurs when Rggobi tool is selected from the menu without any layer loaded and qgis crashes. The error is fixed one hour later and new revision 0.113 is put in the repository. Half an hour later, he confirmed with another mail that the bug is fixed.


Issues - sda4pp - Project ...

code.google.com/p/sda4pp/issues/list?can=1&q=&colspec=ID+Type+Status+Priority+Milestone+Owner+Summary

vkepoglu@gmail.com | My favorites ▾ | Profile | Sign out

sda4pp
Spatial Data Analysis for Point Pattern

Search projects

Project Home | Downloads | Wiki | Issues | Source | Administer

New issue Search All issues for Search Advanced search Search tips

Tip: Type ? for issue tracker keyboard shortcut help.

Select: All None Actions... 1 - 12 of 12 **List** | Grid

ID	Type	Status	Priority	Milestone	Owner	Summary + Labels
1	Defect	Fixed	Medium	----	vkepoglu	› related with Rggobi tool: When selecting ggobi from the menu, without any layer loaded, I get the following error.
2	Defect	Fixed	Medium	----	vkepoglu	› repo: problem with pyc files: plugin compiled on Linux was crashing on Windows
3	Defect	Fixed	Medium	----	vkepoglu	› running SDA4PP in Mac OSX with rpy2: python path issue
4	Defect	WontFix	Low	----	vkepoglu	› incompatibility with spqr which use rpy: Only one R object may be instantiated per session
5	Defect	Fixed	Medium	----	vkepoglu	› related with Density Tool: no numeric colon found for point layer
6	Defect	Done	Medium	----	----	installation's problem
7	Defect	Done	Medium	----	----	win32api
8	Defect	Done	Medium	----	----	R library
9	Defect	Done	Medium	----	----	R libraries
10	Defect	Done	Medium	----	----	libraries:rggobi; iplots
11	Defect	WontFix	Medium	----	----	Kernel Smoothed Intensity
12	Defect	New	High	----	----	Plugin doesn't work with PostGIS layers

1 - 12 of 12 CSV

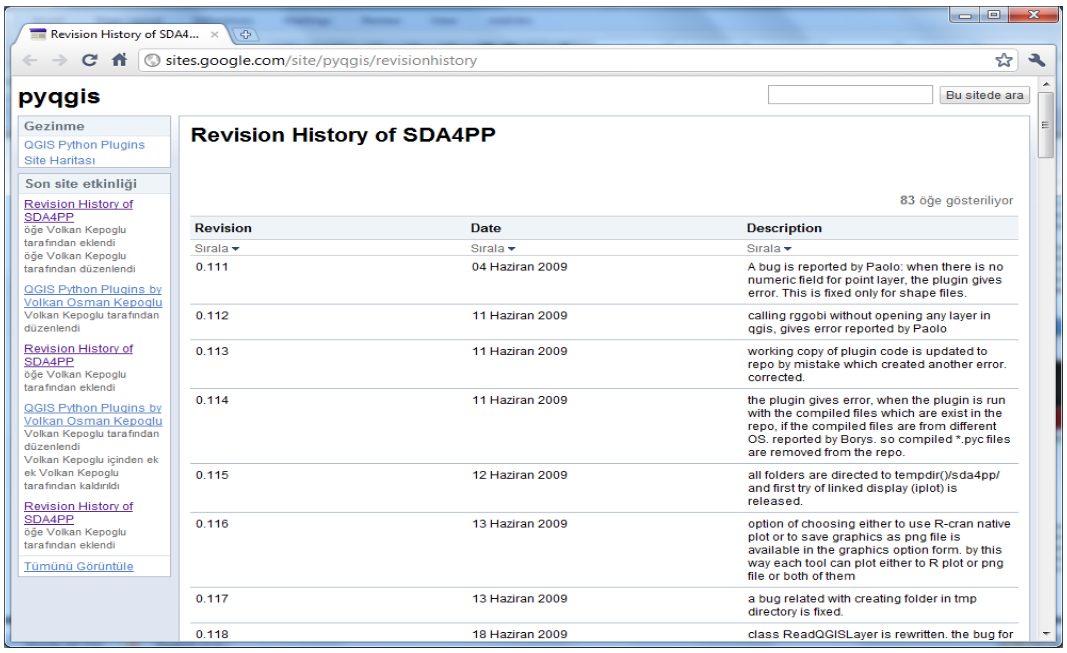
Figure 5. 14. List of Recorded Bugs

Some bugs are directly entered by the user. For example, Luca Scoz reported an error as a bug in December 15, 2009 which is related with coordinate reference system. After several commenting in the bug form, the solution is found. The problem is related with the layer which is defined with wrong coordinate system. For this reason, the plug-in gives an error that this coordinate system is not conformant for this layer. Actually this error is not a bug, instead of this, it is a false usage done by the user. Other bugs are related with how to define “R_HOME” variable as system variable, add the path of R and Python to the path variable and install the R packages and Python modules. Storing reported errors as a bug in the web helps to identify the similar problems for different users.

Sometimes, users do not make any search about the errors that they have faced while installing the SDA system. Instead of using the installer of SDA system, some users have chosen to install the system by themselves, however, the installer setups the path variables. When the related path variables are not correctly setup, the system could not be worked. How to setup the path variables as system variable is explained several times in the QGIS user e-mail list and in the bug system as well. Explaining how to fix the same error several times could be boring. However, when the same error is asked in the e-mail lists and were not replied in a short time by the author of the thesis, some of the members of QGIS community explained the solution again to the users. This has become a kind of self growing support system between the members of the community.

Utility commands can be called with Python language at the terminal level, which makes possible to process some specific tasks automatically like revising the plug-in, converting Qt files to Python syntax and increasing the number of version in the definition of repository file. Numbering format of the version of the plug-in is configured as suitable for doing such automatic process. The first version of the plug-in has started with 0.001. In the next revision, the number is increased by one. The final version has reached to number 0.196. This version numbering format has helped to automate revising the plug-in. To achieve this purpose, a Python module named as Release Manager is developed in order to execute several tasks automatically. This module adds all icons to resource file and converts all GUI files of Qt to Python syntax as well as resource file. And then, it selects all required files like the source code of developed Python modules, resource file of Qt, converted Python version of resource file, log file, Python GUI of each tool, and compresses them as zip file. It also reads the version of the plug-in from __init__ Python file, and increments by one in the definition of repository file.

Another link of web site is the revision history of SDA4PP page as shown in Figure 5.15. The reasons of revising the plug-in are explained in this web page. This page is actually a customizable web form which is provided by sites.google.com. This form displays the texts as a list within the format of several user specified columns. The version number, published date and description for the reasons of revising the plug-in are documented within this format. By this way, the users have a chance to follow what is changed and added in the plug-in.


Revision History of SDA4...
sites.google.com/site/pyqgis/revisionhistory
Bu sitede ara

pyqgis

Gezinme
QGIS Python Plugins
Site Haritası

Son site etkinliği
Revision History of SDA4PP
öğe Volkan Kepoglu tarafından eklendi
öğe Volkan Kepoglu tarafından düzenlendi
QGIS Python Plugins by Volkan Osman Kepoglu
Volkan Kepoglu tarafından düzenlendi
Revision History of SDA4PP
öğe Volkan Kepoglu tarafından eklendi
QGIS Python Plugins by Volkan Osman Kepoglu
Volkan Kepoglu tarafından düzenlendi
Volkan Kepoglu içinden ek ek Volkan Kepoglu tarafından kaldırıldı
Revision History of SDA4PP
öğe Volkan Kepoglu tarafından eklendi
Tümünü Görüntüle

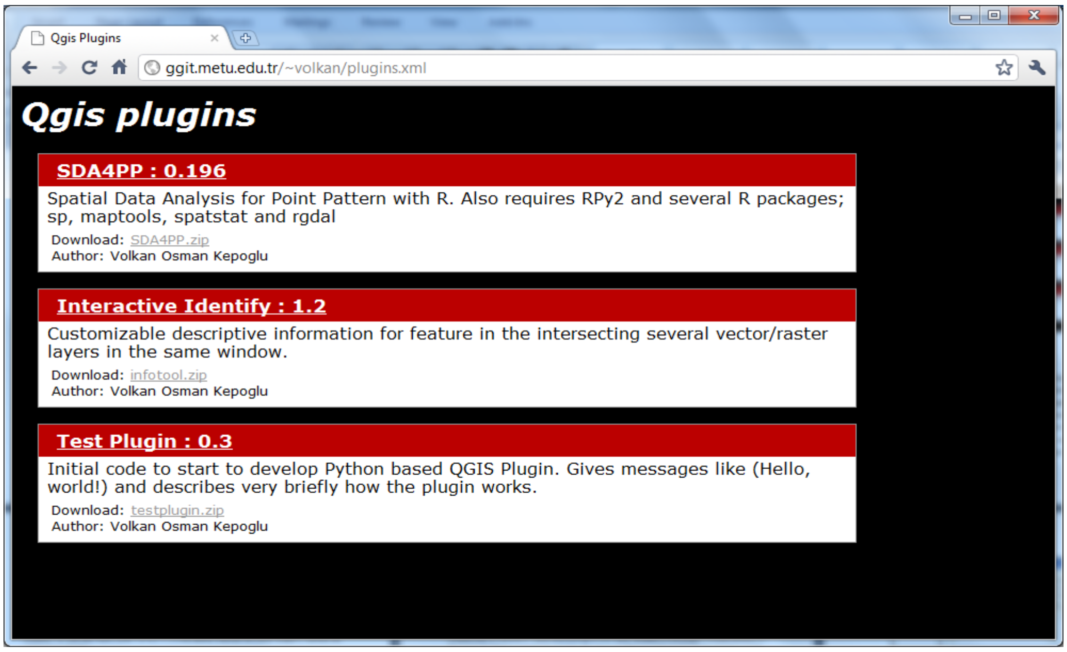
Revision History of SDA4PP

83 öğe gösteriliyor

Revision	Date	Description
Sırala	Sırala	Sırala
0.111	04 Haziran 2009	A bug is reported by Paolo: when there is no numeric field for point layer, the plugin gives error. This is fixed only for shape files.
0.112	11 Haziran 2009	calling rggobi without opening any layer in qgis, gives error reported by Paolo
0.113	11 Haziran 2009	working copy of plugin code is updated to repo by mistake which created another error. corrected.
0.114	11 Haziran 2009	the plugin gives error, when the plugin is run with the compiled files which are exist in the repo, if the compiled files are from different OS. reported by Borys. so compiled *.pyc files are removed from the repo.
0.115	12 Haziran 2009	all folders are directed to tempdir()/sda4pp/ and first try of linked display (iplot) is released.
0.116	13 Haziran 2009	option of choosing either to use R-cran native plot or to save graphics as png file is available in the graphics option form. by this way each tool can plot either to R plot or png file or both of them
0.117	13 Haziran 2009	a bug related with creating folder in tmp directory is fixed.
0.118	18 Haziran 2009	class ReadQGISLayer is rewritten. the bug for

Figure 5. 15. Revision History of the Plug-in

Another link to web site of the plug-in is the repository web page as shown in Figure 5.16. The source code of the plug-in version 0.068 is published in the Python plug-in repository in June 2008. Since then, all Python codes of plug-in can be accessed any time by the QGIS users. Repository is an XML file formatted with extensible stylesheet programming language. In the repository, the name, version, description, download link for compressed zip file and author of the plug-in are defined. Specified version number in the repository and in the initial Python module of plug-in (__init__.py) should match with each other, because the plug-in installer needs consistent version number between two resources while installing the plug-in.


Qgis Plugins

ggit.metu.edu.tr/~volkan/plugins.xml

Qgis plugins

SDA4PP : 0.196

Spatial Data Analysis for Point Pattern with R. Also requires RPy2 and several R packages; sp, maptools, spatstat and rgdal

Download: SDA4PP.zip
Author: Volkan Osman Kepoglu

Interactive Identify : 1.2

Customizable descriptive information for feature in the intersecting several vector/raster layers in the same window.

Download: infotool.zip
Author: Volkan Osman Kepoglu

Test Plugin : 0.3

Initial code to start to develop Python based QGIS Plugin. Gives messages like (Hello, world!) and describes very briefly how the plugin works.

Download: testplugin.zip
Author: Volkan Osman Kepoglu

Figure 5. 16. Python Plug-in Repository of the Plug-in

The compressed zip file should contain the source code of Python files, but not the binary Python files. If Python binary codes are added to the repository which can create an error for Linux users if these files are interpreted in the Windows OS. For this reason, the Python source code of the plug-in as in the compressed zip file should be distributed to have cross platform support in the repository.

The source code of XML file as an example for explaining the structure of the plug-in repository is shown in Figure 5.17. Several plug-in can be published under one repository. In this code, two plug-in, named as SDA4PP and Interactive Identify, are defined. The same structure is repeated twice with the different parameters. First line is the declaration of XML version and the file name written with extensible stylesheet programming language. The plug-in properties should be defined within “plugins” tag (Line 2 and 19). Declaration of one plug-in starts within the enclosed tag of “pyqgis_plugin”. This is defined in Line 3 by specifying the name and version of the plug-in. Description of the plug-in as “description” tag (Line 4), home page of web site address of the plug-in as “homepage” tag (Line 5), minimum version of QGIS that the plug-in requires to work as “qgis_minimum_version” tag (Line 6), the name of compressed zip file as “file_name” tag (Line 7), the developer of the plug-in as “author_name” tag (Line 8), and lastly, web address for downloading the compressed zip file as “download_url” tag (Line 9) are defined as the properties of

Python QGIS plug-in. Similarly, each property is defined from Line 12 to 17 in the same manner for other plug-in, named as Interactive Identify.

```
1        <? Xml version="1.0"?><? Xml-stylesheet
                                      type="text/xsl" href="plugins.xsl"?>
2        <plugins>
3          <pyqgis_plugin name="SDA4PP" version="0.196">
4            <description>Spatial Data Analysis for Point Pattern with R. Also
requires RPy2 and several R packages; sp, maptools, spatstat & rgdal
   </description>
5            <homepage>http://ggit.metu.edu.tr/~volkan/index.html</homepage>
6            <qgis_minimum_version>1.0</qgis_minimum_version>
7            <file_name>SDA4PP.zip</file_name>
8            <author_name>Volkan Osman Kepoglu</author_name>
9            <download_url>http://ggit.metu.edu.tr/~volkan/SDA4PP.zip
   </download_url>
10         </pyqgis_plugin>
11         <pyqgis_plugin name="Interactive Identify" version="1.2">
12           <description>Customizable descriptive information for feature in the
intersecting several vector/raster layers in the same window
   </description>
13           <homepage>http://sites.google.com/site/pyqgis/Home</homepage>
14           <qgis_minimum_version>1.0</qgis_minimum_version>
15           <file_name>infotool.zip</file_name>
16           <author_name>Volkan Osman Kepoglu</author_name>
17           <download_url>http://ggit.metu.edu.tr/~volkan/pyqgis/infotool.zip
   </download_url>
18         </pyqgis_plugin>
19       </plugins>
```

Figure 5. 17. Source Code of Plug-in Repository

Nearly in every six months period, the major release of the plug-in has been developed. Major release of the plug-in with version number, release date and used version of QGIS is shown in Table 5.3. During the development of plug-in in 2.5 years, the version of QGIS is changed from 0.9.2 to 1.5.0. As of February 2011, the latest version of QGIS is 1.7.0. The plug-in is still working with the latest version of the QGIS. Third major release of the plug-in has become more stable with the release of QGIS version 1.0.2. The number of written Python code lines according to the major release of the plug-in is shown in Table 5.4. Grand total of code line of the plug-in has reached to 12,835 lines with the latest version 0.196.

Table 5. 4. Development Progress of Plug-in according to Major Releases

Major Release	**1**	**2**	**3**	**4**	**5**	**Latest**
SDA4PP Version	0.068	0.092	0.120	0.141	0.192	0.196
Release Date	23.06.08	30.01.09	22.06.09	16.11.09	23.06.10	08.09.10
QGIS Version	0.9.2 & 0.10.0	1.0.0 preview2	1.0.2	1.2.0 & 1.3.0	1.4.0	1.4.0 & 1.5.0
Total Code Line	1,993	3,409	5,113	7,958	12,055	12,835

Snapshots of SDA4PP menu according to the major release of the plug-in is also shown in Figure 5.18. Latest revision of SDA4PP plug-in in Windows and Ubuntu OS is shown in Figure 5.19. These snapshots and figures demonstrate how the plug-in is developed in time.

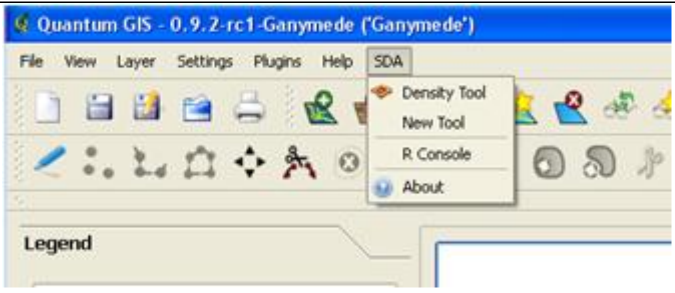
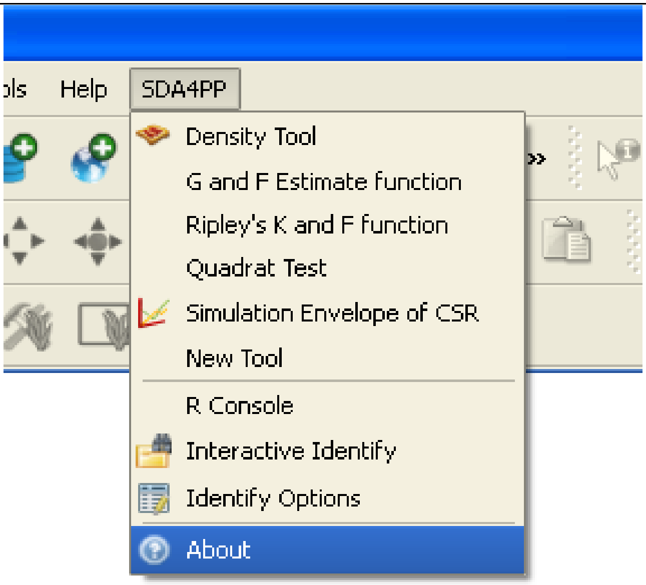
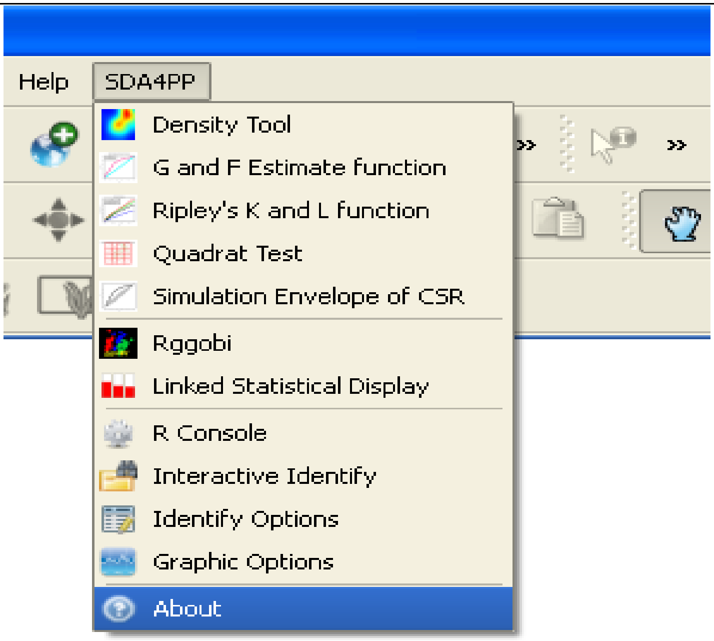
Revisions	Snapshots
First Major Release; 0.068	 Quantum GIS - 0.9.2-rc1-Ganymede ('Ganymede') File View Layer Settings Plugins Help SDA Density Tool New Tool R Console About Legend
Second Major Release; 0.092	 Help SDA4PP Density Tool G and F Estimate function Ripley's K and F function Quadrat Test Simulation Envelope of CSR New Tool R Console Interactive Identify Identify Options About
Third Major Release; 0.120	 Help SDA4PP Density Tool G and F Estimate function Ripley's K and L function Quadrat Test Simulation Envelope of CSR Rggobi Linked Statistical Display R Console Interactive Identify Identify Options Graphic Options About

Figure 5. 18. Major Release of SDA4PP Plug-in

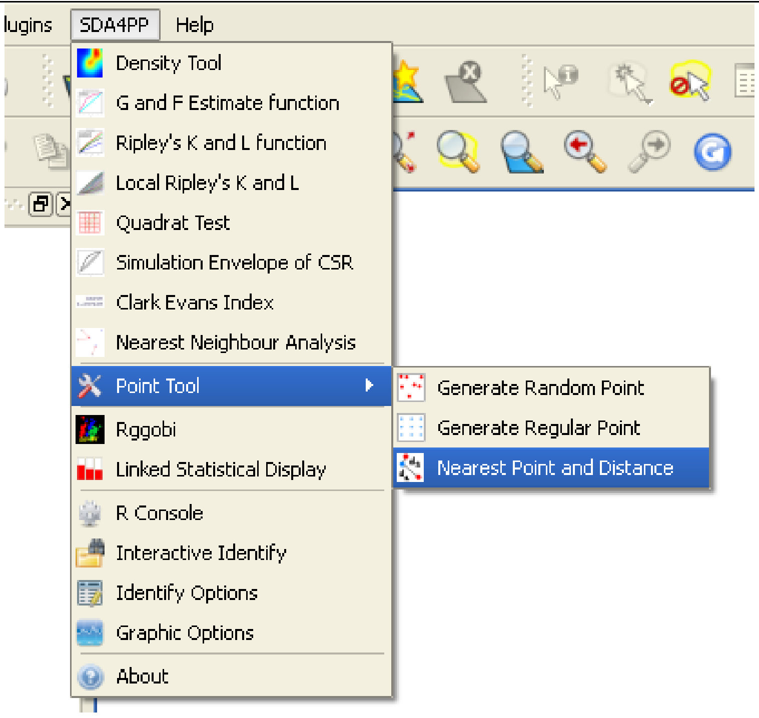
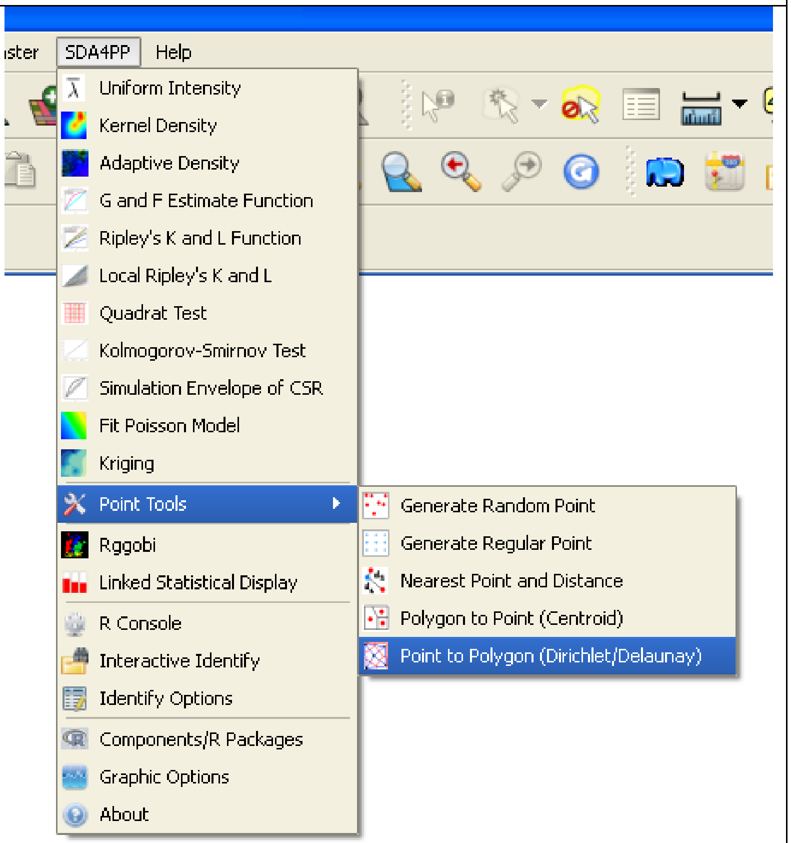
Revisions	Snapshots
Fourth Major Release; 0.141	
Fifth Major Release; 0.192	

Figure 5. 18. Major Release of SDA4PP Plug-in (Continued)

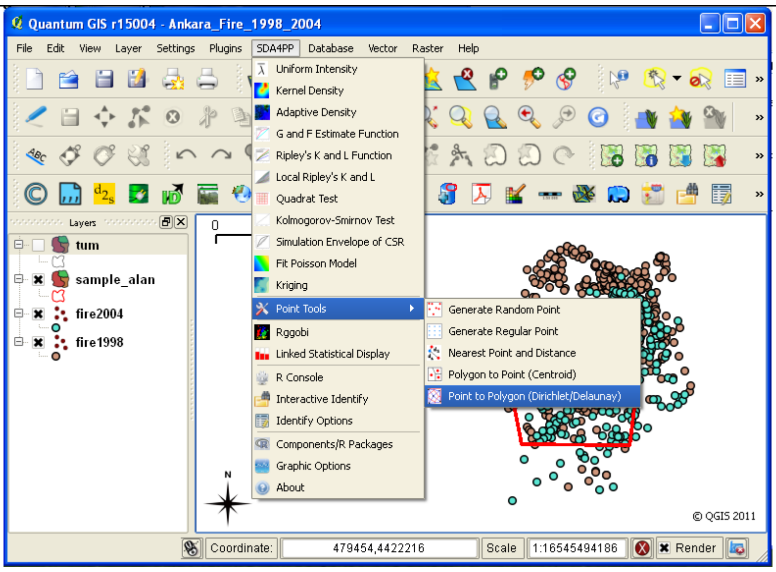
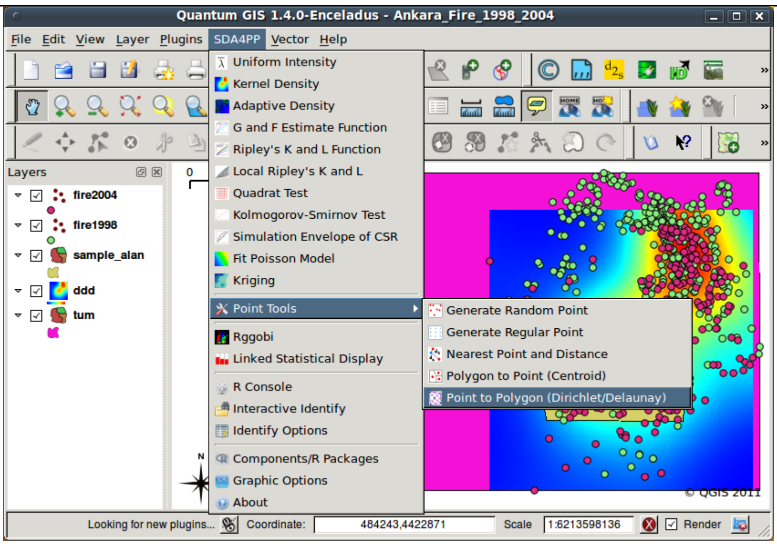
Operating System	Snapshot of Latest Revision; 0.196
In Windows	
In Ubuntu	

Figure 5. 19. Latest Revision of SDA4PP Plug-in in Windows (Top) and Ubuntu (Bottom)

CHAPTER 6

CONCLUSION

6.1. Analysis of SDA System User Statistics

The users of SDA system are researched in order to understand how much interest is seen from the users and who uses the system. The statistics about the user of SDA system is interpreted from two resources; the users who registered themselves by filling a form and the users who entered to the web site of SDA system. Registration statistics are recorded by the web page developed by the author of this thesis. Users' visits statistics are recorded by using Google Analytics that is a free service provided by Google to track the traffic of users for the linked web site. The number of registered users between January 2010 and February 2011 is 127. These users have downloaded the installer of SDA system prepared for Windows operating system 144 times. According to the statistics of Google Analytics recorded in February 2009 and January 2011, the number of absolute unique visitors who visited the web site is nearly 1,600. These figures are explained in detailed in the following sections.

6.1.1. Statistics about Registered Users

A registration form is developed in the web site of SDA system in January 2010. Users are required to fill a registration form in order to install the SDA system from the web site. Two options are available in the registration. The users can prefer either to download the Windows installer or to read the command line installation instructions of Ubuntu distribution.

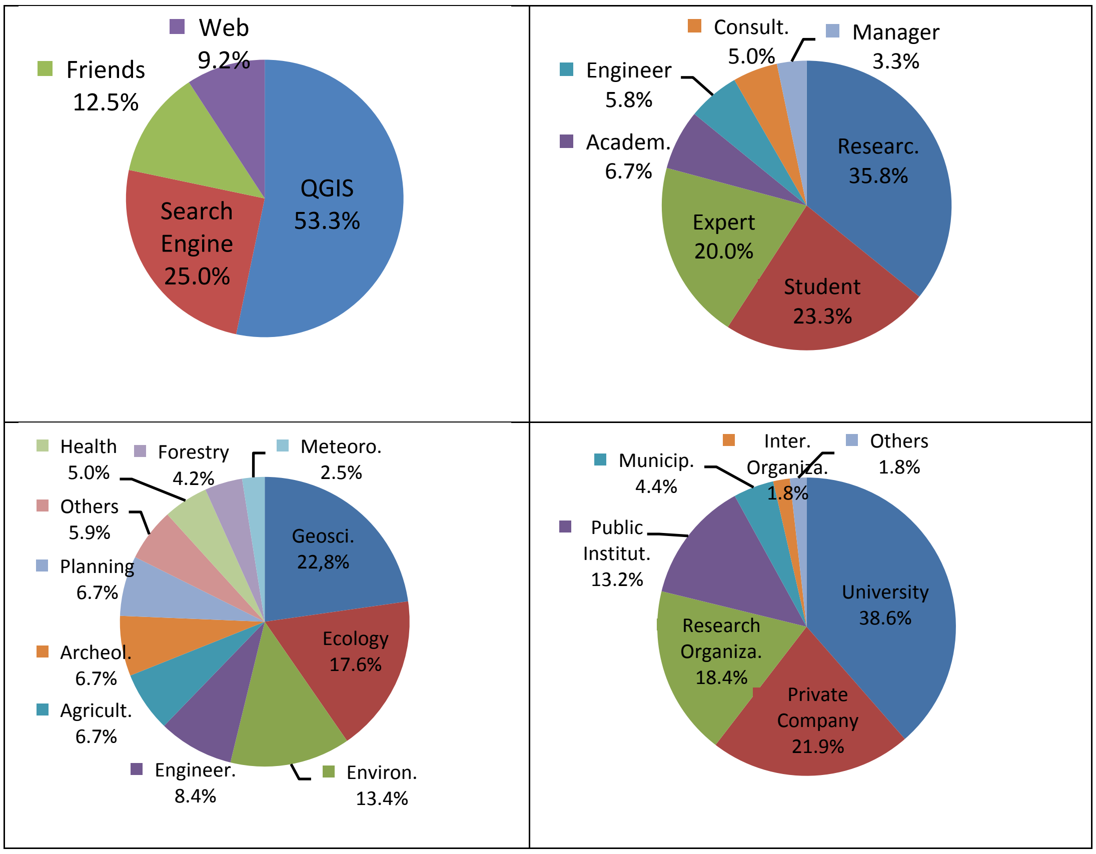


Figure 6. 1. Information about Registered Users; How Heard the SDA System (Top Left), Job Title (Top Right), Application Area (Bottom Left), Organization (Bottom Right)

While users are filling the form, six subjects about the registered users, namely, preferred operating system to install the SDA system, how heard the system, job title, application area, organization and country where the user visited the web site are recorded in nearly one year. There are 127 registered users which are recorded 168 times. 92% (117/127) of the users has preferred to download Windows installer and 8% (10/127) is Ubuntu distribution. Majority of the registered users (53.3%) have heard the SDA system from QGIS resources like forum, Python plug-in repository, e-mail, plug-in installer, web site, 25% of them from web search engine, 12.5% of users from friends, 9.2% of them from directly web (Figure 6.1 – Top Left). Job title of registered users is mainly (65.8%) related with the scientific research like academician, researcher and student. The rest of them are composed of expert (20%), engineer (5.8%), consultant (5%) and manager (3.3%) (Figure 6.1 – Top Right). Registered users' application areas are too wide and differ from in the field of geosciences, ecology, environment, engineering, agriculture, archeology, planning, health and forestry to meteorology (Figure 6.1 – Bottom Left). Variety of application areas is a distinctive property for the SDA field. The same situation can be seen in this figure. According to the figure of organization that the registered user works in, 38.6% of users is from university, 21.9% from private company, 18.4% from public institution, 4.4% from municipality and 1.8% from international organization (Figure 6.1 – Bottom Right). UNICEF, UN and GTZ are some of the examples for most known organizations that exist in the registration records. Registered users are from 34 different countries. Top 10 countries are Italy, France, Turkey, Brazil, United States, India, United Kingdom, Japan, Poland and Portugal (Table 6.1).

Table 6. 1. Registered Users' Countries

Countries	**Frequency**	**Percentage**
Italy	15	11.8%
France	12	9.4%
Turkey	11	8.7%
Brazil	10	7.9%
United States	9	7.1%
India	8	6.3%
United Kingdom	6	4.7%
Japan	5	3.9%
Poland	5	3.9%
Portugal	5	3.9%
Spain	5	3.9%
Germany	4	3.1%
Australia	3	2.4%
Canada	3	2.4%
Others	26	20.5%
Total	127	100.0%

6.1.2. Usage Statistics of Web Site of SDA System

A web site is prepared to inform the users about the development of SDA system. This site is published under the domain of ggit.metu.edu.tr with the following web address; http://ggit.metu.edu.tr/~volkan/index.html. The statistics about the users of SDA system who viewed the web site are recorded by Google Analytics. Several code lines are embedded inside of home page of web site. These codes create a mechanism for Google Analytics service to collect the information about the user who has entered to the site. Statistics are recorded between February 2009 and January 2011. With this linkage mechanism, Google Analytics has provided the statistical figures about the number of absolute unique visitors, visits, average visit per day, countries where the visits come, viewed page number, average spent time in the site by the visitors, returning and new visitors.

Google Analytics (2011) defines absolute unique visitors as "the number of unduplicated visitors to the website in a specified time period like 30 minutes". The visitors are determined using cookies. Total number of absolute unique visitors to the web site of SDA system is 1,593. It is reached to this figure in nearly two years. The distribution of unique visitors per month is shown in Figure 6.2. As it can be seen from the figure, there is an increasing trend and each month the number of visitors has increased. The highest visitor is detected in November 2010.

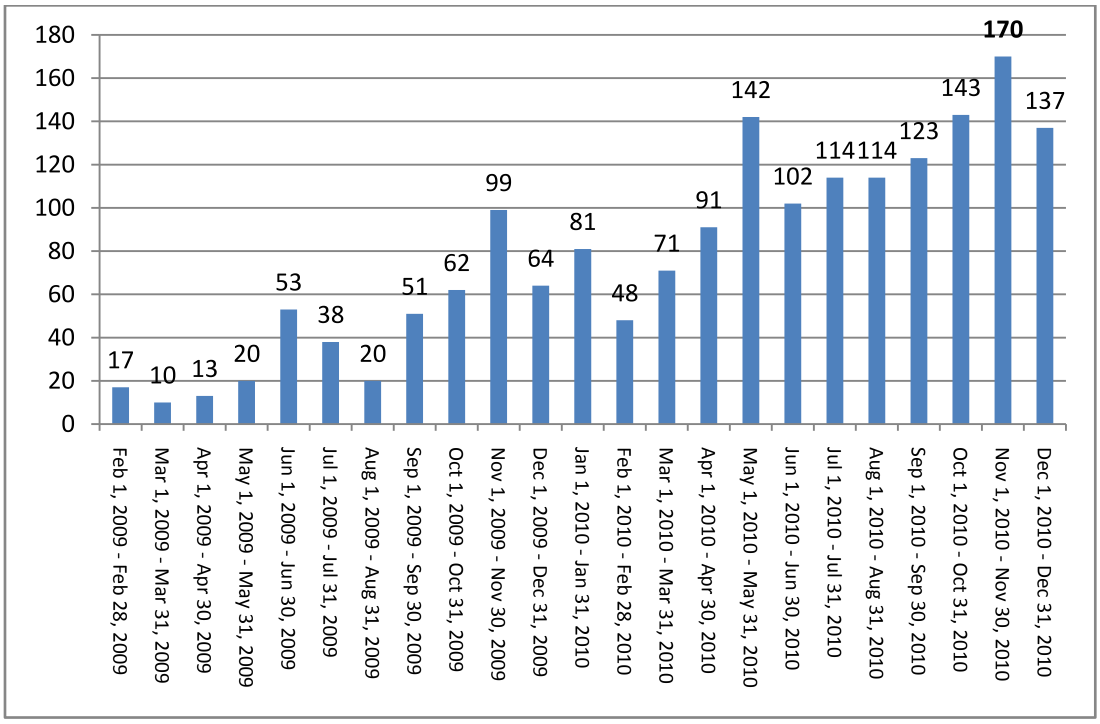


Figure 6. 2. The Number of Absolute Unique Visitors per Month to Web Site of SDA System

1,593 absolute unique visitors visited the site 3,042 times. An average visit per day is 4.16. The distribution of visits per month to the site is shown in Figure 6.3. The number of highest visits per month to the web site of the SDA system is 266. Similarly, the number of visits per month is increasing in time.

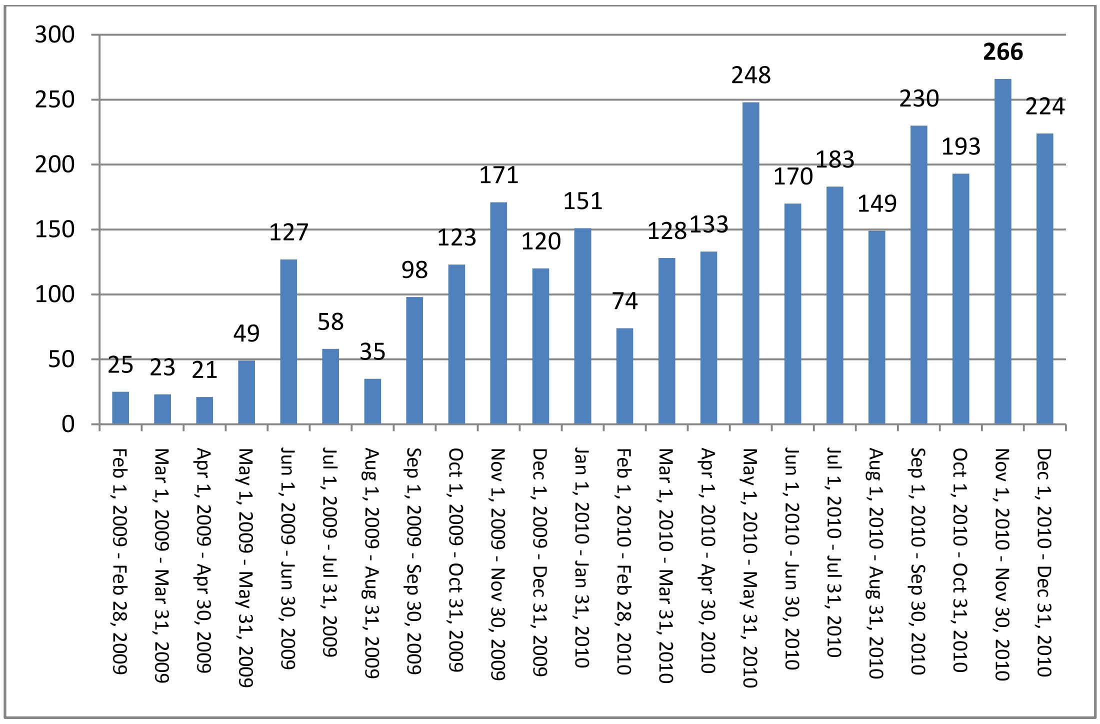


Figure 6. 3. The Number of Visits per Month to Web Site of SDA System

Distribution of the number of visits by top twenty countries is shown in Figure 6.4. The highest visit has come 609 times from Turkey. Top five countries can be listed as Turkey with 609 visits, Italy with 401 visits, United States with 224 visits, Japan with 212 visits and France with 197 visits. 3,042 visits came from 77 countries. The distribution of countries is fully documented in Appendix B and map overlay is shown in Figure 6.5. As it can be seen from the map, the web site of SDA system is used across the world wide.

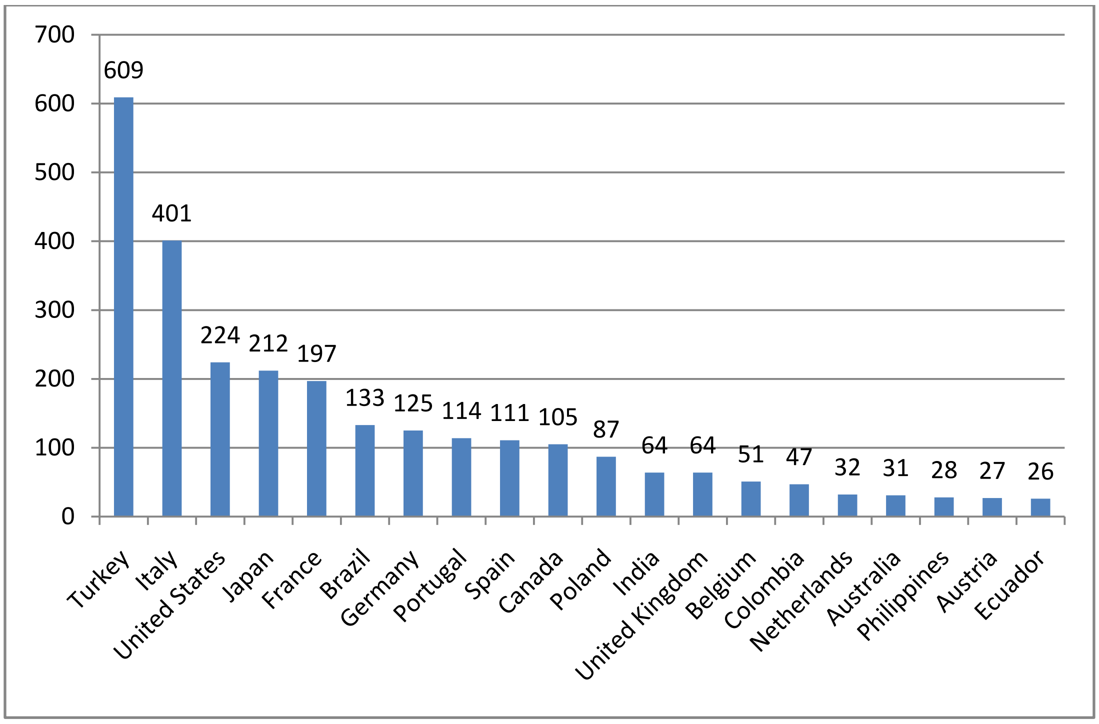


Figure 6. 4. The Number of Visits Coming from Top Twenty Countries

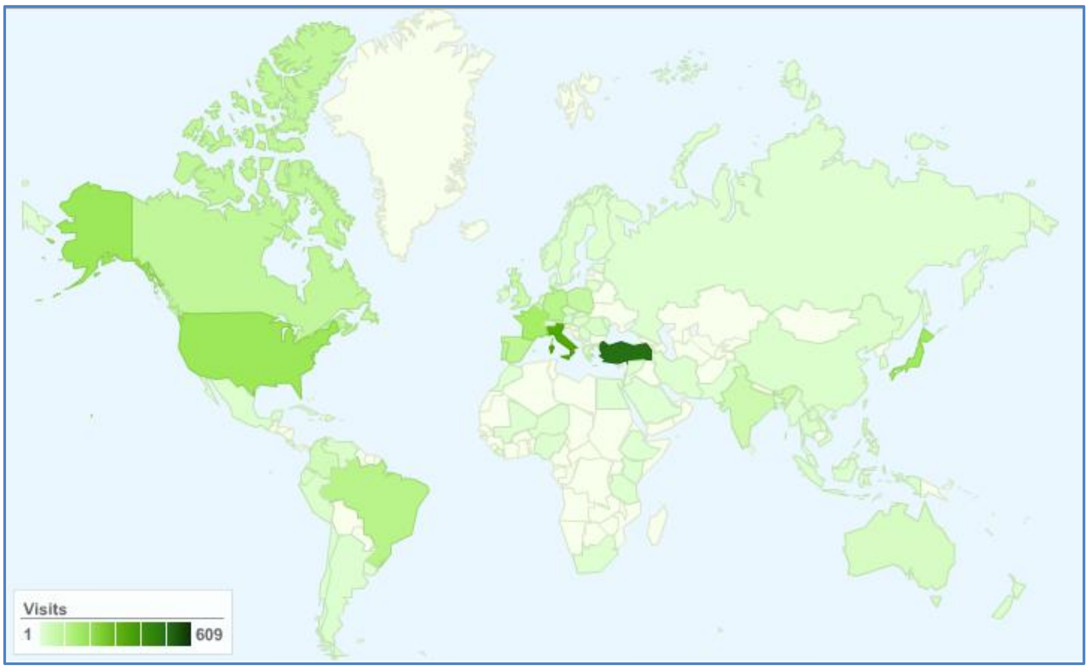


Figure 6. 5. The Number of Visits across the Globe

Google Analytics (2011) defines pageview as an instance of page loaded by browser and adds that the pageview is logged in every time the tracking code is executed. Total number of pageview is 8,574. The distribution of pageview per month to the web site of SDA system is shown in Figure 6.6. An average pageview per visit is 2.82. The range of average pageview per month changes from 1.74 to 4.61.

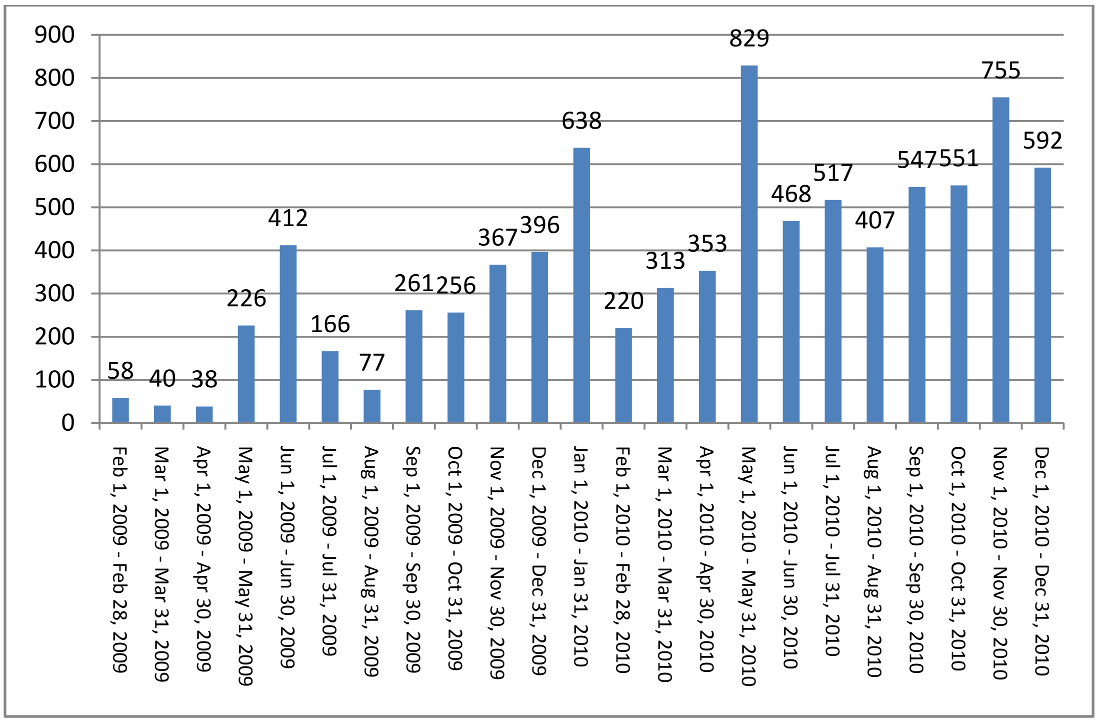


Figure 6. 6. The Number of Pageview per Month in Web Site of SDA System

The average spent time by the visitors in the site is 3:27 minutes. The range of average time on site per month changes from 2:22 minutes to 6:57 minutes. 1,177 of 3,042 (39%) visits came from referring sites, 1,062 of 3,042 (35%) from direct traffic, 803 of 3,042 (26%) from search engines. Google Analytics (2011) defines new visitor as “recorded when any page on the site has been accessed for the first time by a web browser” and returning visitor as “recorded when the cookie exists on the browser accessing the site”. 1,578 of 3,042 (52%) visits is done by a new visitor, while 1,464 of 3,042 (48%) belongs to a returning visitor. Nearly half of the visitors have looked at the site more than once. Mostly used operating system during visits to the site belongs to Windows. 2,076 of 3,042 (68%) operating system is Windows, 669 of 3,042 (22%) is Linux, 254 of 3,042 (8%) is Macintosh, 28 visits’ operating system are not set, 11 visits is iPad and 4 visits made from other operating systems. Mozilla Firefox is the mostly used web browser during visits to the web site of SDA system. Used web browsers during visits to the site with their number of visits are Firefox with 1,752 of 3,042 (58%), Internet Explorer with 497 of 3,042 (16%), Chrome with 483 of 3,042 (16%), Safari with 177 of 3,042 (6%), Opera with 66 of 3,042 (2%) and others with 67 of 3,042 (2%).

6.2. Discussion and Conclusion

The aim of this thesis is to develop a fully coupled SDA with GIS in FLOSS environment. In order to achieve this purpose, a FLOS SDA system is developed by using FLOSS products and by following FLOSS development methodology. GIS users have started to request more SDA tools in their GIS software at the last decade. Until 2004, none of SDA tools are implemented in ESRI ArcGIS Desktop. The lack of SDA functionality in GIS software is the main motivation of this thesis to start a development of SDA system. In this thesis, within 2.5 years, it is succeeded to integrate 14 SDA tools (including exploratory techniques) in FLOS GIS software. The contributions of developing SDA software in the FLOSS environment can be listed as follows:

- Disseminating SDA techniques more widely and easily to GIS users,
- Providing an user friendly FLOS GIS software doing SDA techniques for GIS users without requiring any programming,
- Providing an environment for SDA theorists, researchers and developers a chance to test, control and modify the algorithm of SDA techniques, and to add more SDA tools within GIS environment,
- Developing well documented SDA library from the source code of analysis techniques,
- Sharing SDA knowledge with GIS users,
- Providing interoperable and cross-platform software,
- Achieving FLOS software development.

Used FLOSS products are QGIS as main GIS component of the system, Qt as development framework, Python as programming language, R as SDA component of the system, GDAL and PROJ.4 as two geospatial libraries, and Python bindings for QGIS, Qt and R. SDA system is developed between January 2008 and June 2010. During 2.5 years period, nearly 13,000 Python code lines are written and 14 SDA tools are implemented in the system. The outcome of the development of

FLOS SDA system is creating a plug-in in the main window of QGIS. The plug-in is named as SDA4PP which is the abbreviation of Spatial Data Analysis for Point Pattern. The plug-in is revised 196 times with 5 major releases. In nearly every 6 months a new major version is released.

Development is carried out according to FLOSS development methodology. In general, this method requires publishing the source code of the development, giving information about how the development is ongoing, and attracting the users and developers to help the development process by adopting each person as a member of the project. The source code of the plug-in is published in the repository in June 2008. Since then, all source codes in the repository can be accessible through the Internet. This openness is sustained by forming a Python plug-in repository under the domain of ggit.metu.edu.tr. In order to give information about how the development is ongoing, a dedicated web site is prepared. In this site, each implemented tool is explained briefly, and the snapshots of the tools with the result of the analysis are also shown. In addition, the reasons of new version of the plug-in are explained, bugs can be reported by the users, and news about the development is announced in the blog. In order to attract more developers, the development environment of the system is clarified with their used version of each component. An installer for Windows OS is provided for users to download and install the system more easily.

The statistics about the users of SDA system who viewed the web site are recorded by Google Analytics. According to these statistics, 1,593 absolute unique visitors visited the web site of SDA system between February 2009 and January 2011. In summary, 1,593 visitors came from 77 countries, 3,042 visits, 8,574 page views have recorded nearly in two years. As it can be seen from these statistics, the web site of SDA system is used across the world wide.

Nearly 1,600 visitors have followed the results of the development from the web site of SDA system within a short time like two years. However, this statistic is not a true figure for the usage of plug-in; because this figure belongs to the users who visited web site of the SDA system, but, the figure does not show how many times the plug-in is downloaded and installed. There is no figure about this subject, because repository is in the XML file and Google Analytics could not record the connection to an XML file, but counts the visits that are done to HTML coded web pages. The number of connection to plug-in should be much more than 1,600 users, because there is no need to look at the web site of the SDA system in order to install the plug-in. As it is explained previously, there is a fetching mechanism between the plug-in repository and the user inside of the QGIS. This mechanism downloads the source code of the plug-in from Python plug-in repository and installs the plug-in automatically without entering any HTML page in the site. In other words, the users who installed the plug-in but not additionally looked at the site are not recorded by the Google Analytics. For this reason, it is assumed that the user of plug-in is much higher than 1,600. These figures show that coupling of SDA in GIS takes an attention from many users. The development of SDA system has also seen an interest from the members of the QGIS community. The members of the community were interested with the development of SDA4PP plug-in. They reported bugs and errors, requested new features and tools, replied plug-in users' help messages and accepted the plug-in repository as external author repository in their project.

Integration of SDA in GIS could not be succeeded in the past. Although some of the reasons have out-dated, the integration is still not progressed in that pace. At present, GIS is at the advanced stage as compared in 1960s and 1970s, and computation power is enough to perform all SDA techniques. Although there are many desktop GIS applications, the number of SDA tools in these applications is not enough as compared with the number of GIS tools. GIS vendors' motivations are not still in the scientific research; because the support of SDA market is not strong enough to convince the GIS vendors to couple complete SDA techniques in their GIS products. However, GIS users have started to request more SDA tools in their GIS software. Several standalone SDA software products like Info-Map, SpaceStat and GeoDa are developed as pioneer example applications in order to disseminate and facilitate the SDA field, but, all of them are designed to work only in closed source environment which is subject to break down in their structure as changes occur in operating system in every two years. The cost of upgrading the structure of standalone SDA software could not be met from the market. For this reason, many of them have become out-dated in time with their antiquated architecture.

There is still a gap in the provision of standalone SDA software. This gap is tried to be filled with the development of GeoDa which is a million granted projects. The sustainability of software has broken with Windows Vista in March 2007 after two and half years from the release of stable version in September 2004 due to applying misleading software development methodology in terms of providing sustainability of standalone SDA software. In order to fix the breakdown of software architecture, a new development work is carried out, but the stable version of new GeoDa could not be finished in August 2010. However, the sustainability of the development could not be established with only the development of new version of GeoDa, whose architecture is replaced by adopting FLOS GUI library in order to have cross platform support. An active community should be created around the FLOSS project. If this community is formed in the FLOSS environment by applying FLOSS development methodology, the sustainability of standalone SDA software could be guaranteed. When the scientific SDA research community is grasped around the FLOSS project, the development and the maintenance of the GIS software specialized for SDA field can be continued by the community. The chance to sustain the availability of GIS software fully coupled with SDA techniques in FLOSS environment is higher than closed source if the project has succeeded to create a community supporting the development of software, since the development of software can be done with collaboratively and free of charge when several enthusiasts SDA researchers and developers have adopted to contribute the project.

The commercialization of FLOSS came up at the last decade. This situation has showed a rapid progress, reached to many users from different countries and in each day support of this environment is growing. For example, the use of Linux has spread all over the world and the number of Linux distributions formed by different organizations has exceeded couple of hundreds. This thesis' motivation arises from the development progresses in the FLOSS environment. Development progresses in the geospatial field of FLOSS environment can create an opportunity of window for the development of SDA system in order to facilitate and disseminate the SDA field. Available FLOS spatial libraries and applications are mature enough to develop the SDA system. The number of spatial libraries and applications are increasing everyday within the FLOSS environment.

The major utility of developing SDA software in FLOSS environment should not be only providing the software free but also sharing the SDA knowledge among the researchers coming from different fields. The most predominant factor of SDA field is that many application areas have contributed to the development of SDA field. Many areas have caused to develop different style of analysis. This kind of variety is one of the drawbacks of SDA which requires developing the software working with many researchers coming from different application areas. This collective development work can be carried out easily especially when the chance to read, modify and redistribute the source code of SDA techniques is possible during the development process of the software. Also, this opportunity gives the flexibility of testing the algorithms of SDA tools and provides an environment to discover new SDA techniques for both developers and researchers. Therefore, accessing, studying and distributing the source code of SDA technique is a predominant factor at the coupling of SDA with GIS. Openness of the source code is granted as a right for everyone only in the FLOSS environment. The opportunity to look at the source code of the analysis can also help to learn the analysis among more researchers, avoid reinventing the SDA techniques, implement the technique from existing algorithms and accelerate the teaching of the GIS users.

The freedoms provided by free software are legitimized with free licensees. The invaluable advantage of QGIS is to have these freedoms. The freedoms that QGIS provides as the development tool for the developer, the price advantageous and the simplicity for easy to use GIS software that provide for their users are superior which makes QGIS valuable application as much as commercial proprietary desktop GIS software product for the development of SDA system.

6.3. Further Recommendations

This thesis has proved that the coupling of SDA with GIS could be done by using FLOS GIS software products. Nearly 13,000 line Python code is written in 2.5 years, distributed within the repository via Internet, shared with the QGIS community, tested and used by more than 1,600 users. At the beginning of development, it is not expected to reach that amount of users. This number can be increased by attracting more developers and users with facilitating more collaborative software development environment such as setting up version control system, which makes possible for many programmers to develop the code together, using e-mail lists dedicated to the development, not concentrating on one application area of SDA field, making announcements and advertisements about the project. Especially not limiting the implementation of SDA tools according to one subject of the SDA field could attract more users. SDA system should have many tools including from other application areas of SDA field like spatially continuous data analysis, areal data analysis and spatial interaction data analysis. For example, although there are variety forms of kriging tool in the proprietary GIS software, this tool is requested to be implemented in the plug-in by the community, because it is stated that there is no good implemented example for kriging tool in the FLOSS environment. Adding more popular analysis tools could attract more users to use the developed system. How to contribute to the project should be directly explained to newcomers in the web site. In the FLOSS projects, the contribution can be in the form of code contributions, bug fixes, bug reports, contributing to manuals and documentation, making translation and packaging, preparing tutorials and wiki pages, supporting other users on e-mail lists and forums, and financially sponsoring and funding. All forms of contributions should be organized in the web site by setting up related FLOS tools.

It is recommended to consider that not only the development of SDA system should be done in the FLOSS environment but also the development should be managed as FLOSS project, because continuity of the software development in the FLOSS environment mostly depends on the formation of the community. Some of the mentioned FLOS collaborative methods like setting up a collaborative repository and dedicated e-mail list, making advertisement are not applied during this thesis's development, because the main motivation of this work is to make a development, but not to consider managing the FLOSS project and creating a community around the project. Also, required time for creating a community around the project is much longer than 2 years. It is guessed that between 5 and 10 years could be required to realize this task. Therefore, it should be considered that before starting to improve the developed SDA system of this thesis as the FLOSS project, a couple of SDA enthusiastic developers should be collaborated and they should be ready to spend their spare time both on the development of software and the management of FLOSS project for between 5 to 10 years. FLOSS project requires a long term support to be a known product. For example, QGIS has started to be developed in 2002. After 9 years, the project has reached to be downloaded nearly 100,000 times, whereas GeoDa has reached to 52,000 in 7.5 years, but, GeoDa is not an FLOSS project and QGIS is in the GIS field. The figures give an overall idea about how much effort should be spent in the development of SDA system to be a well known product in the FLOSS environment.

Forming a community around the project is important in terms of providing sustainability for the development of SDA system. Whether more than 1,600 visitors can be accepted as community for this development or not is not clear. This uncertainty can be tested with an announcement to QGIS community stating that it is time to hand over the development of SDA4PP plug-in to requesting developer. Whether any developer will own the development and continue to develop the system is unknown, but this kind of transition mechanism is possible in any time, because source code of the development can be accessible from the Internet. This kind of sustainability has established as a mechanism from the beginning in the FLOSS environment. This is the chance of providing continuity of the development of SDA system in the FLOSS environment.

More than 1,600 users in 2.5 years prove that there is still wide interest for the coupling of SDA in GIS in the FLOSS environment. It is assumed that complete integration of SDA with GIS could be done in the FLOSS environment with a community, when more professional FLOSS development methodology is followed like using appropriate FLOS collaborative tools, making announcements and advertisements, and designing the web site to invite the users and developers to work together. It is hoped for the near future that the complete integration of SDA in GIS can be achieved in the FLOSS environment by creating an active and worldwide supported SDA community. The creation of a SDA community around the FLOSS project should be carried out in order to guarantee the sustainability of SDA system. It can be concluded from the development experiences that are learned while developing the SDA system in the QGIS environment that the development could be achieved in the FLOSS environment with FLOS tools, but organizing a community around the FLOSS project requires setting up a collaborative environment and is a matter of different culture then carrying out the software development.

Another recommendation is to inform that both the developers and users have a right to study the source code of SDA techniques in the FLOS spatial products. The source code of the tool is hundred percent true instructions for understanding how the technique is working. It could be required to examine the source code of SDA techniques because the explanations of SDA techniques are either poorly documented or explained by emphasizing too much mathematical sophistication. The right of examining the source code of SDA tool can help the researchers and developers to understand the SDA techniques more easily, test the techniques, and even to develop new techniques by trying different algorithms. The freedoms that are legitimized by the FLOS licensees match with the merit of scientific research especially required for the SDA field. For example, R is one of FLOS statistical package, which provides great resource and invaluable knowledge about the SDA field. Although the learning curve of R is very steep, it is worth to learn this programming language and environment for the researchers who are interested in the geo-computation area of SDA field.

As a final recommendation, FLOS desktop GIS software products should not be compared with proprietary software in terms of the number and capability of features that they provide. Lack of functionality in FLOS GIS applications is valid in the desktop GIS area. However, in other application areas like programming API's, libraries, spatial databases and web mapping, the quality of FLOSS products are comparable or even superior then the proprietary products. This thesis is interested in the area of desktop GIS. QGIS as the FLOS desktop GIS software product needs many improvements and requires much longer development period in order to have full richness of commercial proprietary desktop GIS software. However, the power of QGIS comes from GNU GPL which provides all freedoms that come with free software. QGIS is a development tool for the developers, has the price advantageous as compared with the commercial proprietary desktop GIS software products and provides the simplicity for easy to use GIS software for GIS users as well as non-GIS users.

REFERENCES

Aktaş, V., (2010), Python Nedir? Bir Yazılımcının Günlüğü, http://www.volkanaktas.com/post/Python-Nedir-.aspx, last accessed on August 23, 2010.

Anselin, L., (1992), SpaceStat, a Software Program for Analysis of Spatial Data, National Center for Geographic Information and Analysis (NCGIA), University of California, Santa Barbara, CA.

Anselin, L., (2000a), Computing Environments for Spatial Data Analyses, Journal of Geographical Systems, 2, 201-220.

Anselin, L., (2000b), GIS, Spatial Econometrics and Social Science Research, Journal of Geographical Systems, 2:11-15.

Anselin, L., and Bao, S., (1997), Exploratory Spatial Data Analysis Linking SpaceStat and ArcView, in Fisher M., Getis A., (eds.) Recent developments in spatial analysis, Springer, Berlin Heidelberg, New York.

Anselin, L. and McCann, M., (2009), OpenGeoDa, Open Source Software for the Exploration and Visualization of Geospatial Data, In GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp.550-551, Seattle, Washington, US.

Anselin, L., Syabri, I., and Kho, Y., (2006), GeoDa: An Introduction to Spatial Data Analysis, Geographical Analysis, 38-1: 5-22.

ArcGIS Resource Center, (2007), FAQ: Does ArcGIS support Microsoft Windows Vista, http://resources.arcgis.com/content/kbase?fa=articleShow&d=34020, last accessed on September 3, 2008.

Bailey, T. C., and Gatrell, A. C., (1995), Interactive Spatial Data Analysis, Pearson Education Limited, England.

Bates, J., and Stone, M., (2006), Communicating Many to Many, in Open Sources 2.0: The Continuing Evolution, ed. by Chris Dibona, Danese Cooper and Mark Stone, O’Reilly Media inc., USA.

Behlendorf, B., (1999), Open Sources as a Business Strategy, in Open Sources: Voices from the Open Source Revolution, ed. by Chris Dibona, Sam Ockman and Mark Stone, O’Reilly Media inc., USA.

Bioconductor, (2008), what is Bioconductor? Bioconductor: Open Source Software for Bioinformatics, http://www.bioconductor.org/whatisit, last accessed on September 8, 2008.

Bivand, R. S., (2007), spatial data in R, R-CRAN, http://r-spatial.sourceforge.net/, last accessed on September 25, 2008.

Bivand, R. S., Pebesma, E. J., and Gomez-Rubio, V., (2008), Applied Spatial Data Analysis with R, Springer Science and Business Media, New York.

Bobholz, M. S., (e-mail, March 5, 2007) complains that GeoDa does not working on Vista, http://geodacenter.asu.edu/openspace/2007-March/000978.html, last accessed on January 5, 2011.

Brown, M. C., (2001), Python: The Complete Reference, McGraw-Hill Companies, USA.

Cascadoss, (2007a), Business Models, CASCADOSS, Development of Transnational Cascade Program on Open Source GIS & RS Software for Environmental Applications, Sixth Framework Program, Identification of New Methods of Promoting and Encouraging Transnational Technology Transfer, http://www.cascadoss.eu/en/index.php?option=com_content&task=view&id=60&Itemid=67, last accessed on August 11, 2010.

Cascadoss, (2007b), Desktop Applications, CASCADOSS, Development of Transnational Cascade Program on Open Source GIS & RS Software for Environmental Applications, Sixth Framework Program, Identification of New Methods of Promoting and Encouraging Transnational Technology Transfer, http://www.cascadoss.eu/en/index.php?option=com_content&task=view&id=14&Itemid=14, last accessed on August 11, 2010.

Chambers, J. M., (2008), Software for Data Analysis Programming with R, Springer, New York, USA.

Chopra S., and Dexter S. D., (2008), Decoding Liberation: The Promise of Free and Open Source Software, Taylor & Francis Group, New York, USA.

CrimeStat, (2011a), CrimeStat III Version 3.3, http://www.icpsr.umich.edu/CrimeStat/, last accessed on January 8, 2011.

CrimeStat, (2011b), CrimeStat III Part I: Program Overview, http://www.icpsr.umich.edu/CrimeStat/files/CrimeStatChapter.1.pdf, last accessed on January 8, 2011.

CSISS, (2002), Specialist Meeting on Spatial Data Analysis Software Tools, Center for Spatially Integrated Social Science, http://www.csiss.org/events/meetings/spatial-tools/, last accessed on September 1, 2008.

Cybersource, (2004), Linux vs. Windows Total Cost of Ownership Comparison, Cybersource: Australia's Leading Linux and Open Source Solutions Company, http://www.cyber.com.au/about/linux_vs_windows_tco_comparison.pdf, last accessed on August 17, 2010.

Dalgıç, S., (2007), Özgür Yazılım Nedir? E-bergi, Nisan 2007, ODTÜ Bilgisayar Topluluğu Elektronik Dergisi, http://e-bergi.com/2007/Nisan/Ozgur-Yazilim-Nedir, last accessed on April 9, 2007.

Debian, (2008), Debian Dağıtımları, Debian, http://www.debian.org/releases/, last accessed on September 11, 2008.

Deitel, H. M., Deitel, P. J., Liperi, J. P., Wiedermann, B. A., (2002), Python How to Program, Prentice Hall, New Jersey, USA.

Distrowatch, (2010), Linux Distributions - Facts and Figures, DistroWatch.com 2001 – 2010, http://distrowatch.com/stats.php?section=popularity, last accessed on August 17, 2010.

Dobias, M., (2006), The New Era: QGIS and Python, Quantum GIS Blog, http://blog.qgis.org/?q=node/29#comment-5185, last accessed on August 26, 2010.

Dobias, M., (2008), Python Bindings, QGIS Wiki, http://www.qgis.org/wiki/Python_Bindings, last accessed on August 26, 2010.

ESRI ArcGIS Desktop Functionality Matrix, (2010), Version 9.1; http://www.esri.com/library/fliers/pdfs/arcgis91-functionality-poster.pdf, Version 9.2; http://www.systematics.co.il/gis/Support/arcgis/pdf/arcgis92-functionality-matrix_1206.pdf, Version 9.3; http://www.esri.com/library/brochures/pdfs/arcgis93-functionality-matrix.pdf, Version 9.3.1; http://www.esri.com/library/brochures/pdfs/arcgis931-functionality-matrix.pdf, Version 10.0; http://www.esri.com/library/brochures/pdfs/arcgis10-functionality-matrix.pdf,last accessed on August 8, 2010.

Everitt, B. S. and Hothorn T., (2006), A Handbook of Statistical Analyses Using R, Chapman & Hall/CRC Taylor & Francis Group, USA.

Fischer, M. M., (2006), Spatial Analysis and Geocomputation: Selected Essays, Springer Berlin – Heidelberg.

Fogel, K., (2006), Producing Open Source Software: How to Run a Successful Free Software Project, O'Reilly, USA.

Fotheringham, A. S., and Rogerson, P., (1993), GIS and Spatial Analytical Problems, International Journal of Geographical Information Systems, 7:3–19.

Fotheringham, A. S., and Rogerson, P., (1995), Spatial Analysis and GIS, Taylor and Francis Limited, UK.

Fotheringham, A. S., Brunsdon, C., Charlton, M., (2000), Quantitative Geography: Perspectives on Spatial Data Analysis, SAGE Publications Inc, London.

Free Software Foundation, (2008), The Free Software Definition, GNU Operating System, http://www.gnu.org/philosophy/free-sw.html, last accessed on March 3, 2008.

GDAL, (2011a), GDAL – Geospatial Data Abstraction Library, http://www.gdal.org, last accessed on February 9, 2011.

GDAL, (2011b), FAQ - General, http://trac.osgeo.org/gdal/wiki/FAQGeneral, last accessed on February 9, 2011.

GDAL, (2011c), GDAL/OGR in Python, http://trac.osgeo.org/gdal/wiki/GdalOgrInPython, last accessed on February 9, 2011.

GDAL, (2011d), OGR Simple Feature Library, http://www.gdal.org/ogr, last accessed on February 9, 2011.

GeoDa, (2008a), GeoDa 0.9.5-i: - ESDA with Dynamically Linked Windows, Center for Spatially Integrated Social Science, http://www.csiss.org/clearinghouse/GeoDa/, last accessed on September 2, 2008.

GeoDa, (2008b), User numbers, GeoDa, https://www.geoda.uiuc.edu/geoda/user-numbers, last accessed on September 4, 2008.

GeoDa, (2010a), GeoDa Users across the Globe, ASU: GeoDa Center for Geospatial Analysis and Computation, http://geodacenter.asu.edu/, last accessed on August 3, 2010.

GeoDa, (2010b), Download GeoDa, ASU: GeoDa Center for Geospatial Analysis and Computation, http://geodacenter.asu.edu/software/downloads, last accessed on August 3, 2010.

GISDevelopment, (2008), Free Open Source Software, Geospatial World, http://www.gisdevelopment.net/downloads/opensource, last accessed on September 12, 2008.

Goodchild, M. F., (1987), Spatial Analytical Perspective on Geographical Information Systems, International Journal of Geographical Information Systems, 1,335–354.

Goodchild, M. F., (2000). The Current Status of GIS and Spatial Analysis, Journal of Geographical Systems, 2:5-10.

Goodchild, M. F., Haining, R. P., Wise, S., and 12 others (1992), Integrating GIS and Spatial Analysis: Problems and Possibilities, International Journal of Geographical Information Systems, 6:407–423.

Google Analytics, (2011), Glossary, Analytics Help, Google Analytics, http://www.google.com/support/analytics/bin/topic.py?hl=en&topic=11285, last accessed on March 24, 2011.

Gonzalez-Barahona, J. M., and Robles, G., (2006), Libre Software in Europe, in Open Sources 2.0: The Continuing Evolution, ed. by Chris Dibona, Danese Cooper and Mark Stone, O'Reilly Media inc., USA.

Gözükeleş, İ. İ., (2006), Free and Open Source Software Hackers in Turkey, Free / Open Source Research Community, http://opensource.mit.edu/papers/WP.pdf, last accessed on August 14, 2010.

Gözükeleş, İ., (2004), Özgür/Açık Kaynak Kodlu Yazılım ve Ulusal Yazılım Politikaları, Sendika.org, http://www.sendika.org/yazi.php?yazi_no=313, last accessed on August 14, 2010.

GRASS GIS, (2008a), GRASS: Introduction: What's GRASS, Geographic Resources Analysis Support System, http://www.grass.itc.it/intro/general.php, last accessed on September 5, 2008.

GRASS GIS, (2008b), GRASS: Introduction: First time Users of GRASS, Geographic Resources Analysis Support System, http://www.grass.itc.it/intro/firsttime.php, last accessed on September 5, 2008.

GRASS GIS, (2008c), GRASS: History, Geographic Resources Analysis Support System, http://www.grass.itc.it/devel/grasshist.html, last accessed on September 5, 2008.

Haining, R. P., (1989), Geography and Spatial Statistics: Current Positions, Future Developments, In Macmillan, B., editor, Remodelling Geography, pages 191–203, Basil Blackwell, Oxford.

Haining, R. P., (2003), Spatial Data Analysis: Theory and Practice, Cambridge University Press, UK.

Infonomics**,** (2002), Free/Libre and Open Source Software: Survey and Study, FLOSS Final Report, International Institute of Infonomics and Berlecon Research GMBH, http://www.flossproject.org/report/index.htm, last accessed on August 14, 2010.

Koch, S., (2005), Free/Open Source Software Development, Idea Group Publishing, USA.

Krysa, J., and Sedek, G., (2008), Source Code, in Software Studies ed. Matthew Fuller, the MIT Press, USA.

Langtangen, H. P., (2006), Python Scripting for Computational Science, Springer, Germany.

Levine, N., (2005), CrimeStat III: A Spatial Statistics Program for the Analysis of Crime Incident Locations, http://www.nedlevine.com/nedlevine17.htm, last accessed on January 8, 2011.

Levine, N., (2006) Crime Mapping and the CrimeStat Program, Geographical Analysis, 38-1: 41-56.

Longley, P. A., and Batty, M., (2003), Advanced Spatial Analysis: the CASA book of GIS, ESRI Press, Redlands, California, USA.

Lutz, M., (2001), Programming Python, O'reilly & Associates, Inc., USA.

Maindonald, J., and Braun, W. J., (2006), Data Analysis and Graphics Using R An Example-based Approach, Cambridge University Press, Cambridge, UK.

MapTools, (2008), MapTools.org, http://maptools.org/, last accessed on September 13, 2008.

Matthews, Stephen A., (2002), Introduction to Dynamic Exploratory Spatial Data Analysis using DynESDA, http://www.pop.psu.edu/gia-core/pdfs/gis_rd_02-10.pdf, last accessed on January 5, 2011.

Microsoft, (2011), a history of Windows, Windows, http://windows.microsoft.com/en-US/windows/history, last accessed on March 21, 2011.

Mitchell, T., (2005), Web Mapping Illustrated, O'Reilly Media, Inc., USA.

Moody, G., (2002), Rebel Code: Linux and the Open Source Revolution, New York: Basic Books.

Muenchen, R. A., (2009), R for SAS and SPSS Users, Springer, New York, USA.

Netcraft, (2010), July 2010 Web Server Survey, Netcraft Ltd., http://news.netcraft.com/archives/2010/07/16/july-2010-web-server-survey-16.html, last accessed on August 2, 2010.

Neteler, M., Beaudette, D. E., Cavallini, P., Lami, L., and Cepicky, J., (2008), GRASS GIS, in Open Source Approaches in Spatial Data Handling, ed. by G. Brent Hall and Michael G. Leahy, Springer, Berlin.

Neteler, M., and Mitasova, H., (2008), Open Source GIS: A GRASS GIS Approach, Third Edition, Springer Science and Business Media, New York.

NSF, (2007), Award Abstract #9978058, Center for Spatially Integrated Social Science, USA National Science Foundation, http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=9978058, last accessed on August 1, 2010.

Ohloh, (2010a), Quantum GIS, Ohloh: the Open Source Network, http://www.ohloh.net/p/tatanike, last accessed on August 17, 2010.

Ohloh, (2010b), GRASS GIS, Ohloh: the Open Source Network, http://www.ohloh.net/p/grass_gis, last accessed on August 17, 2010.

OpenGeoDa, (2010a), OpenGeoDa by Anselin, SourceForge: find and develop open source software, http://sourceforge.net/projects/opengeoda/, last accessed on August 3, 2010.

OpenGeoDa, (2010b), Openspace List, Google Groups, http://groups.google.com/group/openspace-list?hl=en, last accessed on August 3, 2010.

OpenGeoDa, (2010c), OpenGeoDa, Project Home, Updates, Google Project Hosting, http://code.google.com/p/opengeoda/updates/list, last accessed on August 3, 2010.

Open Source Initiative, (2010a), The Open Source Definition Introduction, Open Source Initiative, http://www.opensource.org/docs/osd, last accessed on August 6, 2010.

Open Source Initiative, (2010b), Licenses by Name, Open Source Initiative, http://www.opensource.org/licenses/alphabetical, last accessed on August 6, 2010.

OSGEO**,** (2008a), The Open Source Geospatial Foundation, http://www.osgeo.org/home, last accessed on September 21, 2008.

OSGEO**,** (2008b), OSGEO FAQ, The Open Source Geospatial Foundation, http://www.osgeo.org/content/faq/foundation_faq.html, last accessed on September 21, 2008.

OSGEO, (2008c), Quantum GIS Info Sheet, The Open Source Geospatial Foundation, http://www.osgeo.org/content/faq/foundation_faq.html, last accessed on September 21, 2008.

OSGEO, (2011), GDAL/OGR Info Sheet, OSGEO: Your Open Source Compass, http://www.osgeo.org/gdal_ogr, last accessed on February 9, 2011.

OSSIM, (2008), Awesome Image Processing, http://www.ossim.org/OSSIM/OSSIM_Home.html, last accessed on September 16, 2008.

PROJ4, (2011a), PROJ.4 - Cartographic Projections Library, http://trac.osgeo.org/proj, last accessed on February 9, 2011.

PROJ4, (2011b), FAQ: Frequently Asked Questions, http://trac.osgeo.org/proj/wiki/FAQ, last accessed on February 9, 2011.

Python, (2008a), Python Programming Language: Official Website, http://www.python.org/, last accessed on September 20, 2008.

Python, (2008b), General Python FAQ, Python Documentation, http://docs.python.org/faq/general, last accessed on September 20, 2008.

Python, (2008c), Extending and Embedding the Python Interpreter, Python Documentation, http://docs.python.org/extending/index.html, last accessed on September 20, 2008.

R-CRAN, (2008a), The R Project for Statistical Computing, http://www.r-project.org/, last accessed on September 25, 2008.

R-CRAN, (2008b), frequently asked Questions on R, R FAQ, http://cran.r-project.org/doc/FAQ/R-FAQ.html, last accessed on September 25, 2008.

Ramsey, P., (2007), A Survey of Open Source GIS, FOSS4G 2007: Free and Open Source Software for Geospatial 2007, http://2007.foss4g.org/presentations/view.php?abstract_id=136, last accessed on September 14, 2008.

Raymond, E. S., (2001), the Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly, USA.

Rempt, B., (2001), GUI Programming with Python: QT Edition, Commandprompt Inc., http://www.commandprompt.com/community/pyqt/, last accessed on June 24, 2008.

Riverbank, (2008a), PyQt Whitepaper, Riverbank Computing Limited, http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/pyqt-whitepaper-a4.pdf, last accessed on September 19, 2008.

Riverbank, (2008b), what is PyQt, Riverbank Computing Limited, http://www.riverbankcomputing.co.uk/software/pyqt/intro, last accessed on September 19, 2008.

RPy, (2008), A Simple and Efficient Access to R from Python, RPy, http://rpy.sourceforge.net/index.html, last accessed on September 27, 2008.

Quantum GIS, (2008), Quantum GIS User and Installation Guide Version 0.9.0 Ganymede, http://download.qgis.org/doc/user_guide_en.pdf, last accessed on September 17, 2008.

Quantum GIS, (2010a), Python Plug-in Repositories, QGIS Wiki, http://www.qgis.org/wiki/Python_Plugin_Repositories, last Access on August 29, 2010.

Quantum GIS, (2010b), Project Organigram, QGIS Wiki, http://www.qgis.org/wiki/Project_Organigram, last Access on August 16, 2010.

Qt, (2011), Signals and Slots, Qt Reference Documentation, http://doc.qt.nokia.com/4.7/signalsandslots.html, last accessed on February 15, 2011.

Sandred, J., (2001), Managing Open Source Projects: a Wiley tech brief, John Wiley & Sons, Inc., Canada.

Sherman, G. E., (2008), Desktop GIS: Mapping the Planet with Open Source Tools, Pragmatic Bookshelf, China.

SIP, (2008a), what is SIP? Riverbank Computing Limited, http://www.riverbankcomputing.co.uk/software/sip/intro, last accessed on September 22, 2008.

Sourceforge, (2010), Top Downloads – For all time, updated daily, SourceForge.net, http://sourceforge.net/top/topalltime.php?type=downloads, last accessed on August, 9, 2010.

Söderberg, J., (2008), Hacking Capitalism; the Free and Open Source Software Movement, Taylor & Francis, UK.

Summerfield, M., (2008), Rapid GUI programming with Python and Qt; The Definitive Guide to PyQt Programming, Pearson Education, USA.

Stallman, R. M., (2002), Free Software Free Society: Selected Essays of Richard M. Stallman, ed. Joshua Gay, GNU Press, Free Software Foundation Inc., Boston, MA USA.

Steiniger, S., (2008), An Overview of Free & Open Source Desktop GIS, GeoApt LLC, http://www.spatialserver.net/osgis/, last accessed on September 15, 2008.

Swaroop, C. H., (2005), A Byte of Python, E-book, http://www.swaroopch.com/byteofpython/, last accessed on June 23, 2008.

Terraseer, (2010), SpaceStat Extension, http://www.terraseer.com/products_spacestat_extension.php, last accessed on January 5, 2011.

TIOBE, (2010), Programming Community Index for August 2010, TIOBE Software, http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, last accessed on August 17, 2010.

Trolltech, (2008a), About Qt, Qt Reference Documentation (Open Source Edition), Trolltech, http://doc.trolltech.com/4.3/aboutqt.html, last accessed on September 18, 2008.

Trolltech, (2008b), Qt Cross-Platform Application Framework, Qt, http://trolltech.com/products/qt/, last accessed on September 18, 2008.

Trolltech, (2008c), Features, Qt, http://trolltech.com/products/qt/features, last accessed on September 18, 2008.

UEKAE, (2010a), Hakkında, Pardus Özgürlük için, http://www.pardus.org.tr/hakkinda, last accessed on August 17, 2010.

UEKAE, (2010b), Ürünler, Pardus Özgürlük için, http://www.pardus.org.tr/urunler, last accessed on August 17, 2010.

Vardar, M., A., (2006), Qt'ye Giriş, http://qt.comu.edu.tr/dosyalar/malivardar-qt.pdf, last accessed on September 19, 2008.

Venables, W. N., Smith, D. M., and the R Development Core Team, (2008), an Introduction to R, Notes on R: A Programming Environment for Data Analysis and Graphics Version 2.8.1.

Vries, H., Vries, H., and Oshri, I., (2008), Standards Battles in Open Source Software: The Case of Firefox, Palgrave Macmillan, UK.

Warmerdam, F., (2008), The Geospatial Data Abstraction Library, in Open Source Approaches in Spatial Data Handling, ed. by G. Brent Hall and Michael G. Leahy, Springer, Berlin.

WxPython, (2010), what is wxPython, wxPython, http://www.wxpython.org/what.php, last accessed on August 20, 2010.

APPENDIX A

MAIN GIS MODULE OF SDA4PP PLUG-IN

Complete source code of developed main Python module for GIS part of SDA4PP plug-in named as sda4ppReadQGISLayer is shown as below:

```
from PyQt4 import QtCore
from PyQt4 import QtGui
from qgis.core import *
from qgis.gui import *
#handling outputFile: path name, file name and extension
import os, shutil, tempfile
# write to logfile.log
import wl

class ReadQgisLayer():
  def __init__(self, iface, type="point"):
    dt = "ReadQGISLayer-__init__"
    self.iface = iface
    index = {"point":0,"line":1,"polygon":2,"all":-1}
    self.geomIndex = index[type]

  def getLayerList(self, checkNumericField=""):
    dt = "ReadQGISLayer-getLayerList"
    layerNameList = []
    layerNameControl = []
    mapCanvas = self.iface.mapCanvas()
    for i in range(mapCanvas.layerCount()):
      layer = mapCanvas.layer(i)
```

```
if layer.type() == layer.VectorLayer:
  try:
    # for point, line and polygon layers
    if self.geomIndex == -1:
      if checkNumericField == "":
        layerNameControl.append(str(layer.source()))
        layerNameList.append(str(layer.name()))
      else:
        self.provider = layer.dataProvider()
        self.fieldDict = self.provider.fields()
        numericFound = 0
        for i in range(self.provider.fieldCount()):
          if self.fieldDict[i].typeName() in ["Integer","Real"]:
            layerNameControl.append(str(layer.source()))
            layerNameList.append(str(layer.name()))
            numericFound = 1
            break
        if numericFound == 0:
          QtGui.QMessageBox.information(self.iface.mainWindow(),
              "SDA4PP Plugin Error",
              "This tool requires at least one numeric field.\n" +\
              "Please, add one numeric field to the layer.\n\n" +\
              "The layer, '" + unicode(layer.name()) +\
              "', is excluded from the analysis.")
    # for one type of geometry layer
    elif self.geomIndex == layer.geometryType():
      if checkNumericField == "":
        layerNameControl.append(str(layer.source()))
        layerNameList.append(str(layer.name()))
      else:
        self.provider = layer.dataProvider()
        self.fieldDict = self.provider.fields()
        numericFound = 0
        for i in range(self.provider.fieldCount()):
          if self.fieldDict[i].typeName() in ["Integer","Real"]:
            layerNameControl.append(str(layer.source()))
        layerNameList.append(str(layer.name()))
        numericFound = 1
        break
      if numericFound == 0:
       QtGui.QMessageBox.information(self.iface.mainWindow(),
          "SDA4PP Plugin Error",
          "This tool requires at least one numeric field.\n" +\
          "Please, add one numeric field to the layer.\n\n" +\
          "The layer, '" + unicode(layer.name()) +\
          "', is excluded from the analysis.")
   except Exception, e:
    wl.wl("Exception: "+str(e),dt)
    error = "This plugin does not support unicode characters.\n" +\
       "Please, use only ASCII characters in the path and the name of file.\n\n" +\
       "The layer, '" + unicode(layer.name()) + "', is excluded from the analysis."
    QtGui.QMessageBox.information(self.iface.mainWindow(),
     "SDA4PP Plugin Error", error)
 wl.wl("layerNameList: "+str(layerNameList),dt)
 return layerNameList

def updateLayer(self, layerName):
 ## tool: toolrandom, toolregular, toolnear
 dt = "ReadQGISLayer-updateLayer"
 mapCanvas = self.iface.mapCanvas()
 for i in range(mapCanvas.layerCount()):
  layer = mapCanvas.layer(i)
  if layer.type() == layer.VectorLayer:
   if self.geomIndex == -1:
    if layer.name() == layerName:
     self.userLayer = layer
     self.provider = layer.dataProvider()
   elif self.geomIndex == layer.geometryType():
    if layer.name() == layerName:
     self.userLayer = layer
     self.provider = layer.dataProvider()
 return self.userLayer
```

```
def getFileSource(self, layerName):
  dt = "ReadQGISLayer-getFileSource"
  mapCanvas = self.iface.mapCanvas()
  for i in range(mapCanvas.layerCount()):
    layer = mapCanvas.layer(i)
    if layer.type() == layer.VectorLayer:
      if self.geomIndex == -1:
        if layer.name() == layerName:
          self.userLayer = layer
          self.provider = layer.dataProvider()
          self.fieldDict = self.provider.fields()
          if self.provider.name() == "postgres":
            (error, postLayerSource) = self.writeAsSHPfile(\
                                        description = "postgis")
            if error !=0:
              wl.wl("the error is: " + str(error),dt)
              return "error", error, layer.geometryType()
            else:
              (path, file) = self.removeLayerID(postlayerSource)
              wl.wl("path: "+str(path),dt)
              wl.wl("file: "+str(file),dt)
              wl.wl("layer.geometryType: "+str(layer.geometryType()),dt)
              return path, file, layer.geometryType()
          else:
            layerSource = QtCore.QFileInfo(self.userLayer.source())
            (path, file) = self.removeLayerID(layerSource)
            wl.wl("path: "+str(path),dt)
            wl.wl("file: "+str(file),dt)
            wl.wl("layer.geometryType: "+str(layer.geometryType()),dt)
            return path, file, layer.geometryType()
      elif self.geomIndex == layer.geometryType():
        if layer.name() == layerName:
          self.userLayer = layer
          self.provider = layer.dataProvider()
          self.fieldDict = self.provider.fields()
          if self.provider.name() == "postgres":
            (error, postlayerSource) = self.writeAsSHPfile(\
                                          description = "postgis")
            if error !=0:
              wl.wl("the error is: " + str(error),dt)
              return "error", error
            else:
              (path, file) = self.removeLayerID(postlayerSource)
              wl.wl("path: "+str(path),dt)
              wl.wl("file: "+str(file),dt)
              return path, file
          else:
            layerSource = QtCore.QFileInfo(self.userLayer.source())
            (path, file) = self.removeLayerID(layerSource)
            wl.wl("path: "+str(path),dt)
            wl.wl("file: "+str(file),dt)
            return path, file

      def removeLayerID(self, layerSource):
        ## tool: getFileSource
        dt = "ReadQGISLayer-removeLayerID"
        path = str(layerSource.filePath())
        if path.find("|") != -1:
          path = path.split("|")[0]
        file = str(layerSource.baseName())
        wl.wl("path: "+str(path),dt)
        wl.wl("file: "+str(file),dt)
        return path, file

      def getExtentObj(self):
        ## tool: toolregular, toolrandom
        return self.userLayer.extent()

      def getEncodingObj(self):
        ## tool: toolregular, toolrandom
        dt = "ReadQGISLayer-getEncodingObj"
```

```
  encoding = self.provider.encoding()
  wl.wl("encoding: "+str(encoding),dt)
  return encoding

def getProj(self):
  ## tool: density, quadrattest
  dt = "ReadQGISLayer-getProj"
  layerProj = self.userLayer.srs().toProj4()
  wl.wl("layerProj: "+str(layerProj),dt)
  return str(layerProj)

def getProjByPointLayerName(self, layerName):
  ## tool: kriging
  dt = "ReadQGISLayer-getProjByPointLayerName"
  mapCanvas = self.iface.mapCanvas()
  for i in range(mapCanvas.layerCount()):
    layer = mapCanvas.layer(i)
    if layer.type() == layer.VectorLayer:
      if layer.geometryType() == 0:
        if layer.name() == layerName:
          layerProj = layer.srs().toProj4()
          wl.wl("layerProj: "+str(layerProj),dt)
          return str(layerProj)

def getFeatureCount(self):
  ## tool: localKL,
  dt = "ReadQGISLayer-getFeatureCount"
  featureNumber = self.provider.featureCount()
  wl.wl("featureCount: "+str(featureNumber),dt)
  return featureNumber

def getFieldNameList(self, allFieldType="", returnFieldType=""):
  ## tool: density, iplot, kriging
  dt = "ReadQGISLayer-getFieldsNameWithCombo"
  fieldList = []
  fieldNameTypeDict = {}
  # all field types are required for tool: iplot
  if allFieldType == "":
    fieldstype = ["Integer","Real"]
  else:
    fieldstype = ["Integer","Real","String"]
  for i in range(self.provider.fieldCount()):
    ftype = self.fieldDict[i].typeName()
    if ftype in fieldstype:
      try:
        fieldList.append(str(self.fieldDict[i].name()))
        fieldName = str(self.fieldDict[i].name())
        fieldType = str(self.fieldDict[i].typeName())
        fieldNameTypeDict[fieldName] = fieldType
      except Exception, e:
        wl.wl("Exception: "+str(e),dt)
        error = "This plugin does not support unicode characters.\n" +\
                "Please, use only ASCII characters in the field name.\n\n" +\
                "The field name, '" + unicode(self.fieldDict[i].name()) +\
                "', in the point layer, '" + unicode(self.userLayer.name()) +\
                "',\nhas non-english characters." +\
                "This field is excluded from the analysis."
        QtGui.QMessageBox.information(self.iface.mainWindow(),
              "SDA4PP Plugin Error", error)
  # returning field names and types as the dict is required for tool: iplot
  if returnFieldType == "":
    return fieldList
  else:
    return fieldNameTypeDict

def getRasterList(self):
  ## tool: kriging,
  dt = "ReadQGISLayer-getRasterList"
  layerNameList= []
  layerNameControl = []
  mapCanvas = self.iface.mapCanvas()
  for i in range(mapCanvas.layerCount()):
    layer = mapCanvas.layer(i)
    if layer.type() == layer.RasterLayer:
      try:
        layerNameControl.append(str(layer.source()))
        layerNameList.append(str(layer.name()))
      except Exception, e:
        wl.wl("Exception: "+str(e),dt)
        error = "This plugin does not support unicode characters.\n" +\
             "Please, use only ASCII characters in the path and the name of file.\n\n" +\
             "The layer, '" + unicode(layer.name()) + "', is excluded from the analysis."
        QtGui.QMessageBox.information(self.iface.mainWindow(),
          "SDA4PP Plugin Error", error)
  wl.wl("rasterNameList: "+str(layerNameList),dt)
  return layerNameList

def getRasterSource(self, layerName):
  ## tool: kriging,
  dt = "ReadQGISLayer-getRasterSource"
  mapCanvas = self.iface.mapCanvas()
  for i in range(mapCanvas.layerCount()):
    layer = mapCanvas.layer(i)
    if layer.type() == layer.RasterLayer and layer.name() == layerName:
      return layer.source()

def getSelectedDataFromIDField(self):
  ## tool: iplot,
  dt = "ReadQGISLayer-getDataFromSelectedField"
  featureIDlist = []
  if self.userLayer.selectedFeatureCount() > 0:
    features = self.userLayer.selectedFeatures()
    wl.wl("features: "+str(features),dt)
    for feature in features:
      featureIDlist.append(feature.id())
    wl.wl("featureIDlist: "+str(featureIDlist),dt)
    return featureIDlist, self.provider.featureCount()
  else:
```

```
    return "error", "No selection found in the map."

def setSelectionFromQuery(self, recordList, zoom = "True"):
  ## tool: iplot, GFestimate, KLripley
  dt = "ReadQGISLayer-setSelectionFromQuery"
  self.userLayer.setSelectedFeatures(recordList)
  if zoom == "True":
    mapCanvas = self.iface.mapCanvas()
    mapCanvas.setCurrentLayer(self.userLayer)
    # set extent to the extent of our layer
    mapCanvas.setExtent(self.userLayer.extent())
    mapCanvas.zoomToSelected()
    mapCanvas.refresh()
  return 0

def checkFeatureNumberInPolyLayer(self):
  ## tool: density,
  if self.provider.featureCount() == 1:
    return 0, 0
  else:
    if self.userLayer.selectedFeatureCount() == 1:
      (error, selectionLayerSource) = self.writeAsSHPfile(description = \
                              "selected feature in polygon layer",
                                              useSelection = 1)
      if error != 0:
        return error, 0
      else:
        return 0, selectionLayerSource
    else:
      return "Either polygon layer should have only one feature\n" + \
           "Or select only one feature from the polygon layer.", 0

def writeAsSHPfile(self, description = "new", useSelection = 0):
  ## tool: self.checkFeatureNumberInPolyLayer, self.getFileSource,
  dt = "ReadQGISLayer-writeAsSHPfile"
  # trying to found unique shp filename
```

```
notFileExist = 0
counter = 0
file = self.createOutputDir() + str(self.userLayer.name()) + ".shp"
if os.path.exists(file):
  while (notFileExist == 0):
    outputfilename = self.createOutputDir() + \
                  str(self.userLayer.name()) + \
                  "_" + str(counter) + ".shp"
    if not(os.path.exists(outputfilename)):
      file = outputfilename
      break
    else:
      counter = counter + 1
wl.wl("unique filename and path for " + str(description) + \
      " layer to be saved as shp file: "+str(file),dt)
layer2shpFileSource = QtCore.QFileInfo(file)
newSHPfilePath = layer2shpFileSource.filePath()
# writing new layer as shp file
p4s = self.userLayer.srs()
encoding = self.provider.encoding()
if useSelection == 1:
  qgisFeature = self.userLayer.selectedFeatures()
  writer = QgsVectorFileWriter(newSHPfilePath, encoding,
                              self.fieldDict, QGis.WKBPolygon, p4s)
  writer.addFeature(qgisFeature[0])
  error = writer.hasError()
  del writer
else:
  error = QgsVectorFileWriter.writeAsShapefile(self.userLayer,
                                              newSHPfilePath,
                                              encoding, p4s)
if error != 0:
  wl.wl("the error is: " + str(description) + \
        " layer could not be written as SHP file.",dt)
  return "The " + str(description) + \
         " layer could not be written as SHP file.", 0
    wl.wl("layer2shpFileSource: "+str(layer2shpFileSource),dt)
    return 0, layer2shpFileSource

  def createOutputDir(self):
    ## tool: self.writeAsSHPfile, rggobi, toolnear, density, toolrandom,
    ##       toolregular
    dt = "ReadQGISLayer-createOutputDir"
    # creating a writable folder
    tempDir = tempfile.gettempdir()
    if os.name == "nt":
      tempDir = tempDir.replace("\\","/")
    outputpathdir = tempDir + "/SDA4PP/"
    if not(os.path.exists(outputpathdir)):
      os.mkdir(outputpathdir)
    wl.wl("outputpathdir: "+str(outputpathdir),dt)
    return outputpathdir

  def saveShape(self, outputFile):
    ## tool: toolrandom, toolregular, quadrat
    dt = "ReadQGISLayer-saveShape"
    fileInfo = QtCore.QFileInfo(outputFile)
    # add a (hardcoded) layer and zoom to its extent
    vlayer = QgsVectorLayer(fileInfo.filePath(),
                         fileInfo.completeBaseName(), "ogr")
    if not vlayer.isValid():
      QtGui.QMessageBox.information(self.iface.mainWindow(),
          "SDA4PP Plugin Error",
          "Shape file: " + str(outputFile) + " is created.\n" + \
          "It seems that the layer is not valid or could not be added to Qgis.\n" + \
          "The problem can be related with gdal library or python-gdal binding.\n" + \
          "Please check the installation of gdal libraries.")
      return "error"
    addToTOC = QtGui.QMessageBox.question(self.iface.mainWindow(),
        "SDA4PP Plugin", "Shapefile: " + outputFile + \
        "\n\nWould you like to add the new layer to the TOC?",
        QtGui.QMessageBox.Yes, QtGui.QMessageBox.No,
                            QtGui.QMessageBox.NoButton)
    # adding raster layer to TOC: yes
    if addToTOC == QtGui.QMessageBox.Yes:
      QgsMapLayerRegistry.instance().addMapLayer(vlayer)
      self.zoomToLayerExtent(vlayer)
    # adding raster layer to TOC: no
    return 0

  def zoomToLayerExtent (self, layer):
    ## tool: self.saveRaster, self.saveShape
    mapCanvas = self.iface.mapCanvas()
    mapCanvas.setCurrentLayer(layer)
    # set extent to the extent of our layer
    mapCanvas.setExtent(layer.extent())
    mapCanvas.zoomToSelected()
    mapCanvas.refresh()

  def saveRaster(self, outputFile, toolname=""):
    ## tool: adaptive, density, poisson, kriging
    dt = "ReadQGISLayer-saveRaster"
    fileInfo = QtCore.QFileInfo(outputFile)
    # add a (hardcoded) layer and zoom to its extent
    rstlayer = QgsRasterLayer(fileInfo.filePath(),
                            fileInfo.completeBaseName())
    if not rstlayer.isValid():
      QtGui.QMessageBox.information(self.iface.mainWindow(),
          "SDA4PP Plugin Error",
          "Raster file: " + str(outputFile) + " is created.\n" + \
          "It seems that the layer is not valid or could not be added to Qgis.\n" + \
          "The problem can be related with gdal library or python-gdal binding.\n" + \
          "Please check the installation of gdal libraries.")
      return "error"
    addToTOC = QtGui.QMessageBox.question(self.iface.mainWindow(),
       "SDA4PP Plugin", "Raster file: " + outputFile + \
       "\n\nWould you like to add the new layer to the TOC?",
       QtGui.QMessageBox.Yes, QtGui.QMessageBox.No,
                            QtGui.QMessageBox.NoButton)
    # adding raster layer to TOC: yes
    if addToTOC == QtGui.QMessageBox.Yes:
      # default: single band
      if toolname == "":
        rstlayer.setDrawingStyle(QgsRasterLayer.SingleBandPseudoColor)
        rstlayer.setColorShadingAlgorithm(QgsRasterLayer.PseudoColorShader)
        rstlayer.setContrastEnhancementAlgorithm(\
          QgsContrastEnhancement.StretchToMinimumMaximum, False)
      # three bands for kriging tool
      elif toolname == "kriging":
        pass
      # add layer to the registry
      QgsMapLayerRegistry.instance().addMapLayer(rstlayer);
      self.zoomToLayerExtent(rstlayer)
    # adding raster layer to TOC: no
    return 0

  def saveDialog(self, parent, extension = "shp"):
    #     dirName,      filtering,     saveText,      suffix
    dt = "ReadQGISLayer-saveDialog"
    index = {"shp":["UI/lastShapefileDir","Shapefiles (*.shp)",
                  "Save output shapefile","shp"],
           "csv":["sda4pp/csvDir",
                  "Comma Separated Values (*.csv)",
                  "Save output comma separated values","csv"],
           "tif":["sda4pp/tifDir",
                  "Geo-Tagged Image File (*.tif)",
                  "Save output geo-tagged image file","tif"],
           "img":["sda4pp/imgDir",
                  "Erdas Imagine Images (*.img)",
                  "Save output erdas imagine images","img"], }
    settings = QtCore.QSettings()
    dirName = settings.value(index[extension][0]).toString()
    filtering = QtCore.QString(index[extension][1])
    encode = settings.value("UI/encoding").toString()
    fileDialog = QgsEncodingFileDialog(parent, index[extension][2],
                                        dirName, filtering, encode)
    fileDialog.setDefaultSuffix( QtCore.QString(index[extension][3]))
    fileDialog.setFileMode(QtGui.QFileDialog.AnyFile)
    fileDialog.setAcceptMode(QtGui.QFileDialog.AcceptSave)
    fileDialog.setConfirmOverwrite(True)
    if not fileDialog.exec_() == QtGui.QDialog.Accepted:
      return None, None
    files = fileDialog.selectedFiles()
    settings.setValue(index[extension][0], QtCore.QVariant(\
        QtCore.QFileInfo(unicode(files.first())).absolutePath()))
    try:
      (f,e) = str(files.first()), str(fileDialog.encoding())
    except Exception, e:
      wl.wl("Exception: "+str(e),dt)
      error = "This plugin does not support unicode characters.\n" +\
            "Please, use only ASCII characters in the path and the name of file.\n\n" +\
            "The output folder, '" + unicode(files.first()) + \
            "', is changed to default folder:\n" +\
            self.createOutputDir()
      QtGui.QMessageBox.information(self.iface.mainWindow(),
        "SDA4PP Plugin Error", error)
      return None, None
    return f,e

  def doDistanceMatrix(self, layer1, layer2, outPath):
    dt = "ReadQGISLayer-doDistanceMatrix"
    provider1 = layer1.dataProvider()
    provider2 = layer2.dataProvider()
    allAttrs = provider1.attributeIndexes()
    provider1.select(allAttrs)
    allAttrs = provider2.attributeIndexes()
    provider2.select(allAttrs)
    csvList = []
    distArea = QgsDistanceArea()
    inFeat = QgsFeature()
```

```
outFeat = QgsFeature()
inGeom = QgsGeometry()
outGeom = QgsGeometry()
first = True
while provider1.nextFeature(inFeat):
  inGeom = inFeat.geometry()
  inID = inFeat.id()
  if first:
    featList = range(layer2.featureCount())
    first = False
    data = [""]
    for i in range(layer2.featureCount()):
      provider2.featureAtId(int(i), outFeat, True, [-1])
      data.append("'V' + str(outFeat.id())+"")
    csvList.append(data)
  data = [inID]
  for j in featList:
    provider2.featureAtId(int(j), outFeat, True)
    outGeom = outFeat.geometry()
    dist = distArea.measureLine(inGeom.asPoint(), outGeom.asPoint())
    data.append(float(dist))
  csvList.append(data)
csvtxt = ""
for lines in csvList:
  for field in lines:
    csvtxt += str(field) + ","
  csvtxt = csvtxt[:-1] + "\n"
f = open(outPath, "w")
f.write(csvtxt)
f.close()
return 0
```

APPENDIX B

DISTRIBUTION OF VISITS PER COUNTRY

Distribution of visits done by absolute unique visitors in which country they visited the web site of SDA system between February 2009 and January 2011 is shown below in Table B.1.

Table B.1. Distribution of Visits per Country (Statistic Recorded by Google Analytics)

Rank	Country	Visits	Avgerage Time on Site	% New Visits *	Bounce Rate **
1.	Turkey	609	00:03:46	39.74%	49.26%
2.	Italy	401	00:04:29	44.64%	41.40%
3.	United States	224	00:01:56	66.07%	55.80%
4.	Japan	212	00:03:20	41.51%	50.94%
5.	France	197	00:03:26	60.91%	45.18%
6.	Brazil	133	00:03:24	38.35%	54.89%
7.	Germany	125	00:01:50	56.00%	66.40%
8.	Portugal	114	00:04:03	53.51%	42.11%
9.	Spain	111	00:04:16	48.65%	35.14%
10.	Canada	105	00:04:31	71.43%	53.33%
11.	Poland	87	00:02:51	52.87%	58.62%
12.	India	64	00:03:36	79.69%	32.81%
13.	United Kingdom	64	00:03:58	56.25%	51.56%
14.	Belgium	51	00:02:19	47.06%	66.67%
15.	Colombia	47	00:02:45	42.55%	63.83%
16.	Netherlands	32	00:01:59	90.62%	37.50%
17.	Australia	31	00:01:29	58.06%	41.94%
18.	Philippines	28	00:04:08	67.86%	32.14%
19.	Austria	27	00:02:30	59.26%	70.37%
20.	Ecuador	26	00:00:57	15.38%	73.08%
21.	Czech Republic	24	00:04:05	58.33%	41.67%
22.	Romania	24	00:05:12	41.67%	37.50%
23.	Greece	22	00:03:35	63.64%	36.36%

Table B.1. Distribution of Visits per Country (Statistic Recorded by Google Analytics) (Continued)

Rank	Country	Visits	Avgerage Time on Site	% New Visits *	Bounce Rate **
24.	Taiwan	18	00:07:20	61.11%	27.78%
25.	Switzerland	18	00:03:03	61.11%	55.56%
26.	Venezuela	18	00:01:05	55.56%	77.78%
27.	Peru	14	00:03:06	28.57%	50.00%
28.	Norway	13	00:05:35	23.08%	23.08%
29.	Laos	13	00:02:07	76.92%	53.85%
30.	China	12	00:03:50	66.67%	41.67%
31.	Mexico	11	00:02:47	63.64%	45.45%
32.	Hungary	11	00:06:05	63.64%	63.64%
33.	Indonesia	11	00:05:02	72.73%	36.36%
34.	New Zealand	10	00:01:15	90.00%	40.00%
35.	Sweden	10	00:01:09	80.00%	20.00%
36.	Chile	9	00:03:19	55.56%	55.56%
37.	Argentina	8	00:00:24	62.50%	50.00%
38.	South Korea	7	00:01:20	42.86%	71.43%
39.	Ireland	7	00:01:47	57.14%	42.86%
40.	Moldova	7	00:06:14	100.00%	42.86%
41.	Costa Rica	6	00:05:09	50.00%	50.00%
42.	Serbia	6	00:04:14	83.33%	33.33%
43.	Thailand	5	00:07:58	100.00%	40.00%
44.	Nigeria	5	00:00:00	40.00%	100.00%
45.	Russia	4	00:04:42	75.00%	75.00%
46.	Kenya	4	00:00:02	50.00%	75.00%
47.	Uruguay	4	00:07:14	75.00%	25.00%
48.	Vietnam	4	00:00:00	100.00%	100.00%
49.	Finland	3	00:02:45	66.67%	66.67%
50.	Niger	3	00:06:10	66.67%	66.67%
51.	Israel	3	00:08:34	100.00%	0.00%
52.	South Africa	3	00:03:06	100.00%	33.33%
53.	Finland	3	00:02:45	66.67%	66.67%
54.	Iran	3	00:00:00	100.00%	100.00%
55.	Luxembourg	3	00:03:01	100.00%	66.67%
56.	Mali	2	00:00:19	50.00%	50.00%
57.	Malaysia	2	00:00:00	100.00%	100.00%
58.	Myanmar	2	00:06:02	50.00%	50.00%
59.	Estonia	2	00:00:00	100.00%	100.00%
60.	Egypt	2	00:00:00	100.00%	100.00%
61.	Slovakia	2	00:00:00	100.00%	100.00%
62.	Syria	1	00:02:02	100.00%	0.00%

Table B.1. Distribution of Visits per Country (Statistic Recorded by Google Analytics) (Continued)

Rank	**Country**	**Visits**	**Avgerage Time on Site**	**% New Visits ***	**Bounce Rate ****
63.	Slovenia	1	00:00:00	100.00%	100.00%
64.	Morocco	1	00:03:50	0.00%	0.00%
65.	Pakistan	1	00:03:25	100.00%	0.00%
66.	Lithuania	1	00:00:14	100.00%	0.00%
67.	Singapore	1	00:00:00	0.00%	100.00%
68.	Tanzania	1	00:00:00	100.00%	100.00%
69.	Bangladesh	1	00:00:00	100.00%	100.00%
70.	Saudi Arabia	1	00:00:00	100.00%	100.00%
71.	Montenegro	1	00:00:00	100.00%	100.00%
72.	New Caledonia	1	00:00:00	100.00%	100.00%
73.	Cuba	1	00:00:00	100.00%	100.00%
74.	Ethiopia	1	00:00:00	100.00%	100.00%
75.	Denmark	1	00:00:00	100.00%	100.00%
76.	Liberia	1	00:00:00	100.00%	100.00%
77.	Dominican Rep.	1	00:00:00	100.00%	100.00%
TOTAL		3042			

*** Visit** is a period of interaction between a visitor's browser and a particular website, ending when the browser is closed or shut down, or when the user has been inactive on that site for a specified period of time (Default is 30 minutes).

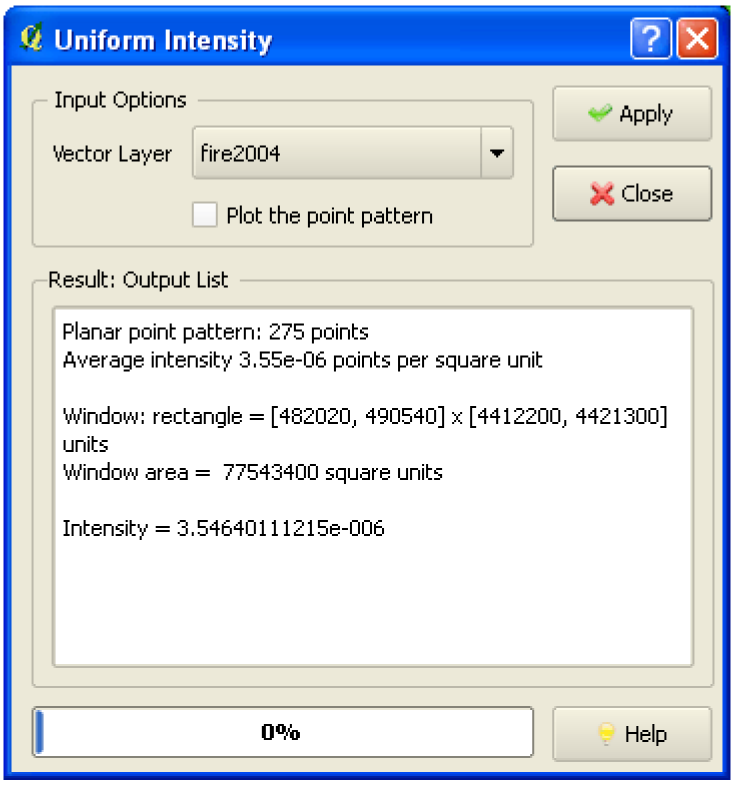
**** Bounce Rate** is the percentage of single-page visits or visits in which the person left your site from the entrance (landing) page.

APPENDIX C

INTEGRATED SDA TECHNIQUES; GUI AND ANALYSIS RESULT

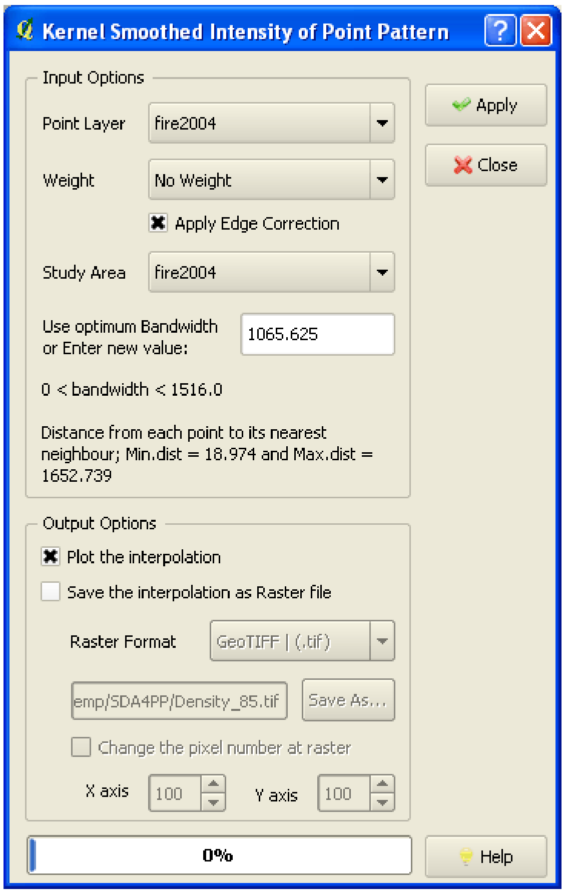
During the development of FLOS SDA system, integrated tools in QGIS are listed with their graphical user interface (GUI) and analysis result as follows:

1. Uniform Intensity: GUI and Analysis Result



```
Planar point pattern: 275 points
Average intensity 3.55e-06 points per square unit

Window: rectangle = [482020, 490540] x [4412200, 4421300]
units
Window area =  77543400 square units

Intensity = 3.54640111215e-006
```

2. Kernel Smoothed Intensity of Point Pattern: GUI



Kernel Smoothed Intensity of Point Pattern

Input Options

Point Layer: fire2004

Weight: No Weight

[✓] Apply Edge Correction

Study Area: fire2004

Use optimum Bandwidth or Enter new value: 1065.625

0 < bandwidth < 1516.0

Distance from each point to its nearest neighbour; Min.dist = 18.974 and Max.dist = 1652.739

Output Options

[✓] Plot the interpolation

[ ] Save the interpolation as Raster file

Raster Format: GeoTIFF | (.tif)

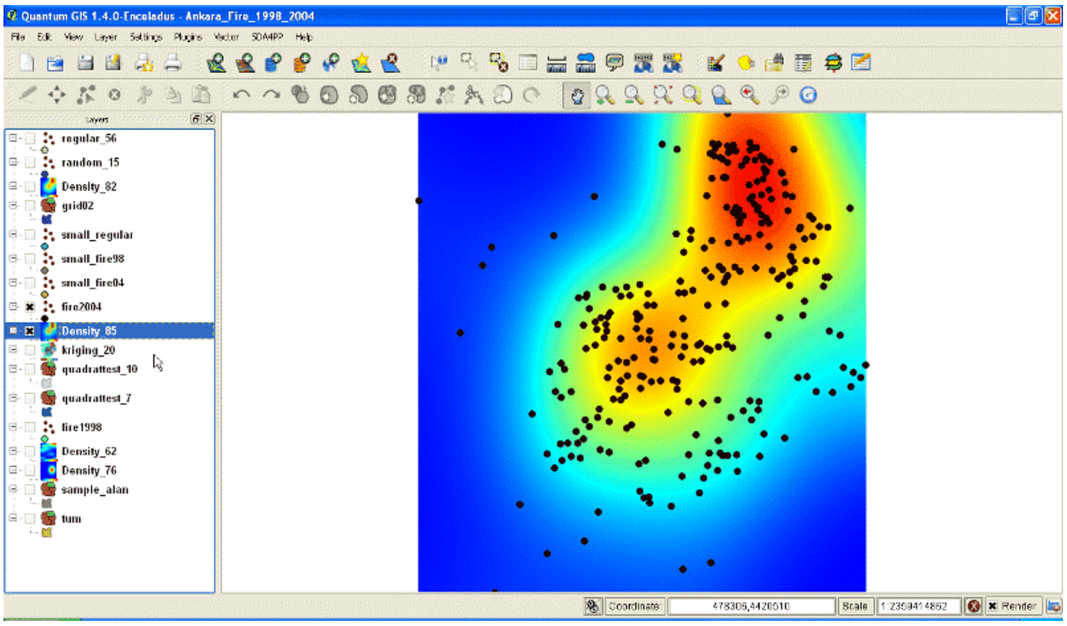
emp/SDA4PP/Density_85.tif

[ ] Change the pixel number at raster

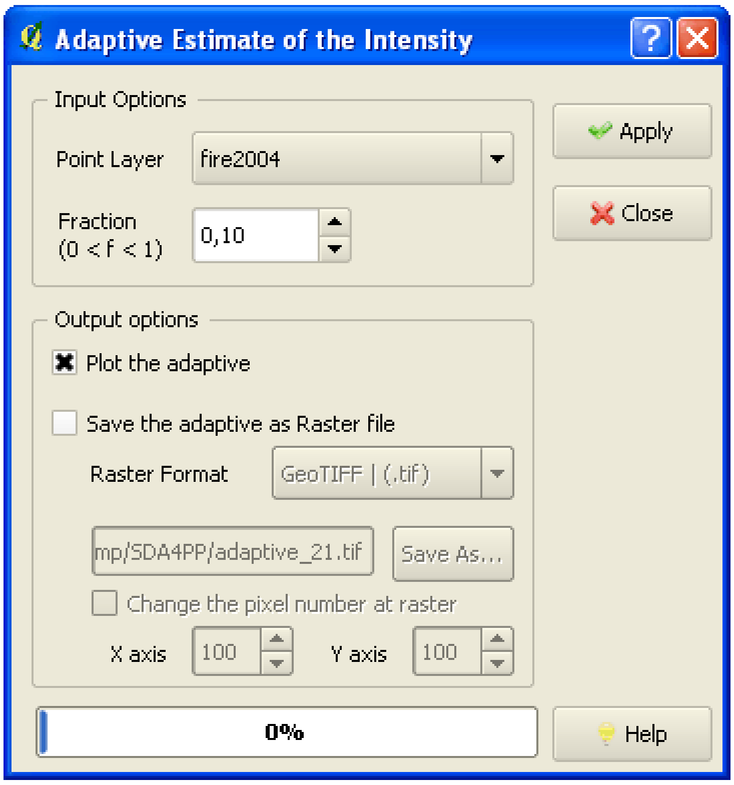
X axis: 100 Y axis: 100

0%

Analysis Result



3. Adaptive Estimate of the Intensity: GUI


Adaptive Estimate of the Intensity

Input Options

Point Layer: fire2004

Fraction (0 < f < 1): 0,10

Output options

[✓] Plot the adaptive

[ ] Save the adaptive as Raster file

Raster Format: GeoTIFF | (.tif)

mp/SDA4PP/adaptive_21.tif

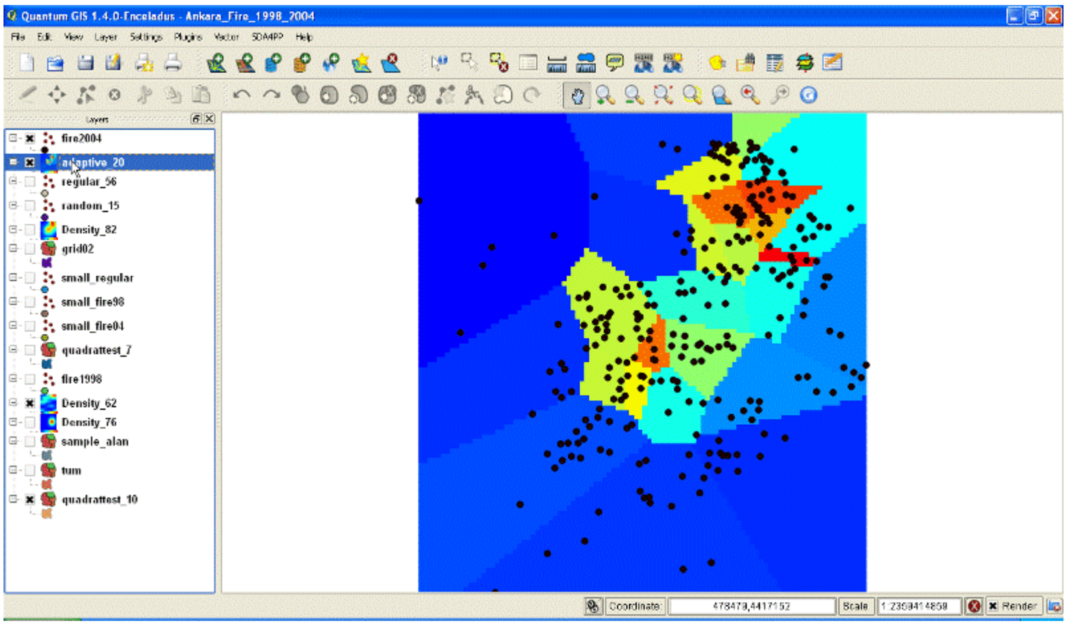
Save As...

[ ] Change the pixel number at raster

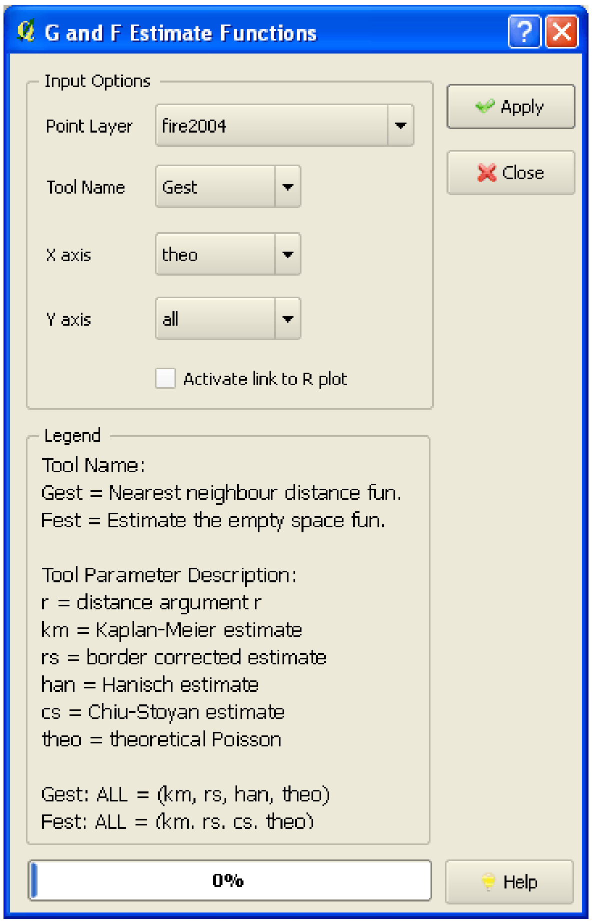
X axis: 100 Y axis: 100

0%

Analysis Result



4. G and F Estimate Functions: GUI



Input Options

Point Layer: fire2004

Tool Name: Gest

X axis: theo

Y axis: all

☐ Activate link to R plot

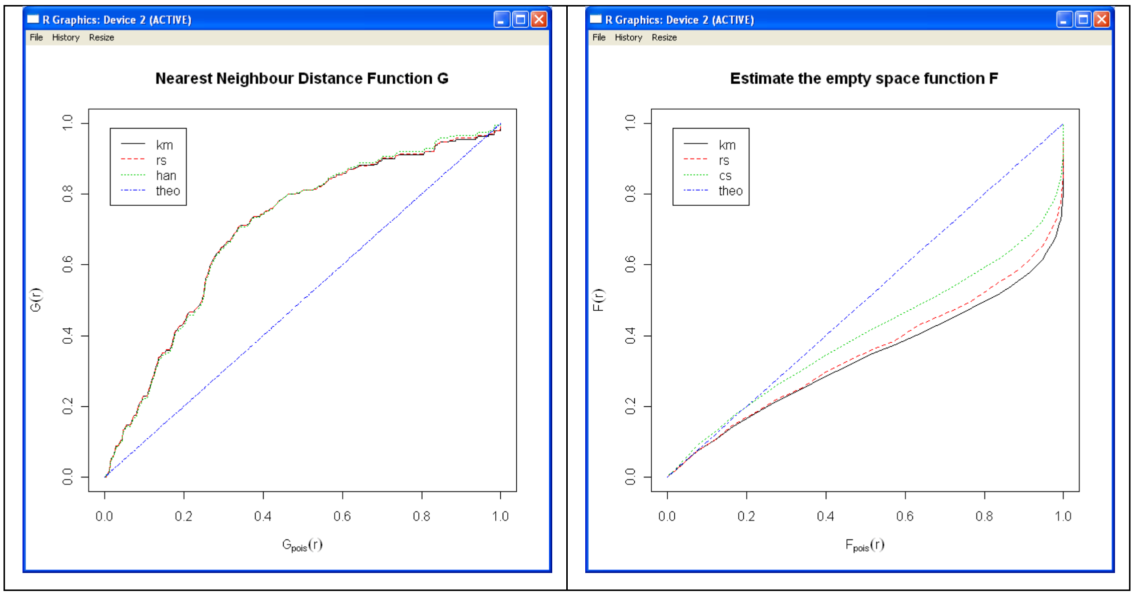
Legend

Tool Name:
Gest = Nearest neighbour distance fun.
Fest = Estimate the empty space fun.

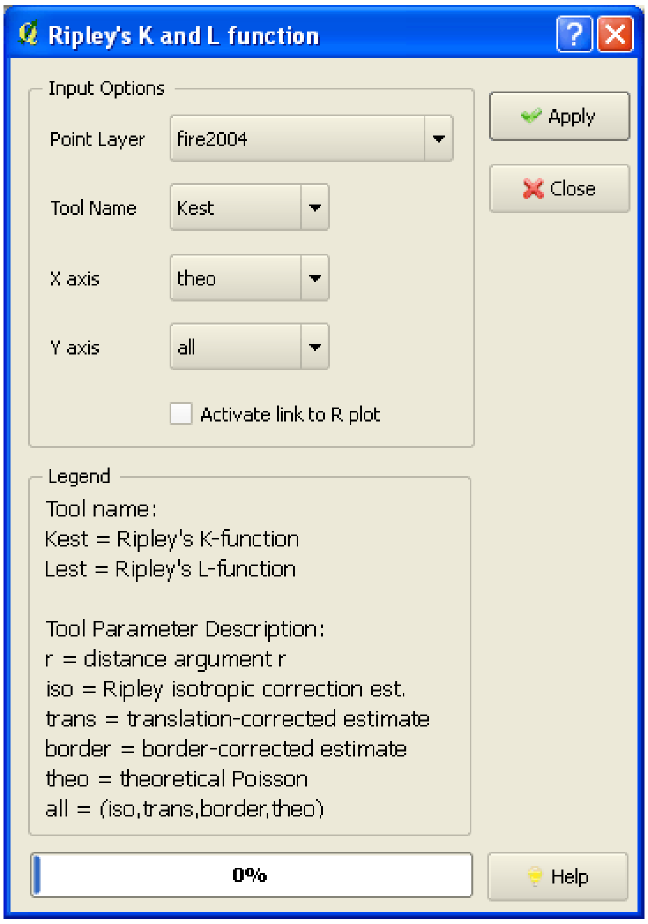
Tool Parameter Description:
r = distance argument r
km = Kaplan-Meier estimate
rs = border corrected estimate
han = Hanisch estimate
cs = Chiu-Stoyan estimate
theo = theoretical Poisson

Gest: ALL = (km, rs, han, theo)
Fest: ALL = (km. rs. cs. theo)

Analysis Results: G (Left Side) and F (Right Side) Estimate Functions



5. Ripley's K and L Functions: GUI



Input Options

Point Layer: fire2004

Tool Name: Kest

X axis: theo

Y axis: all

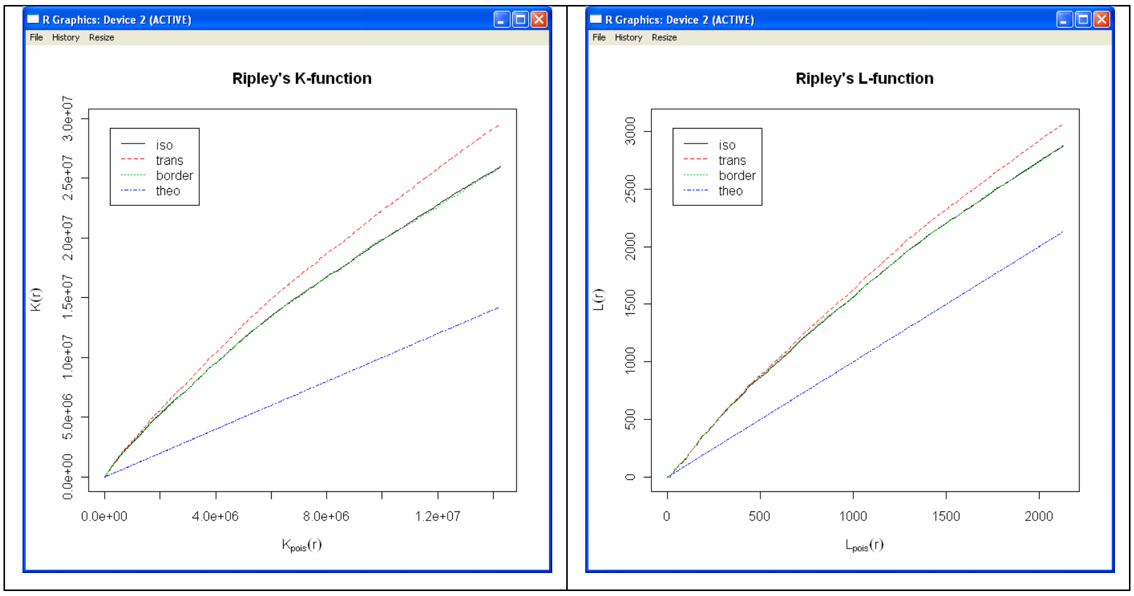
[ ] Activate link to R plot

Legend

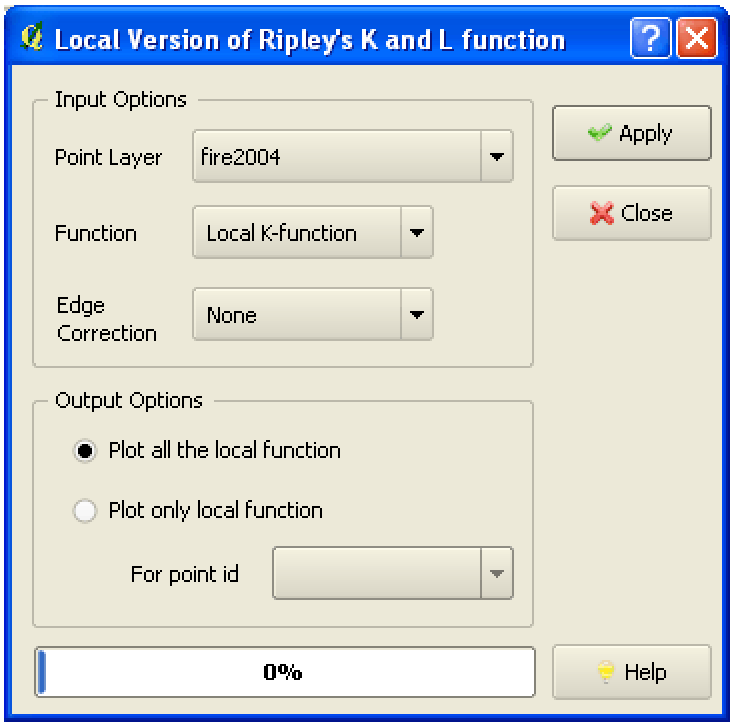
Tool name:
Kest = Ripley's K-function
Lest = Ripley's L-function

Tool Parameter Description:
r = distance argument r
iso = Ripley isotropic correction est.
trans = translation-corrected estimate
border = border-corrected estimate
theo = theoretical Poisson
all = (iso,trans,border,theo)

Analysis Result: Ripley's K (Left Side) and L (Right Side) Functions



6. Local Version of Ripley's K and L Functions: GUI



Input Options

Point Layer: fire2004

Function: Local K-function

Edge Correction: None

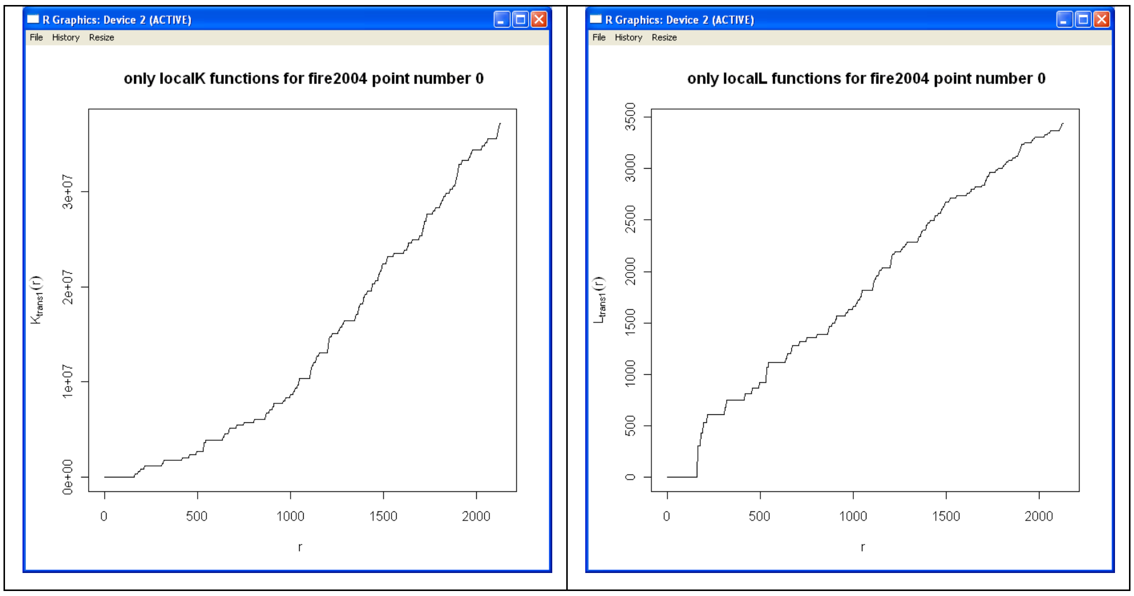
Output Options

[✓ Plot all the local function]

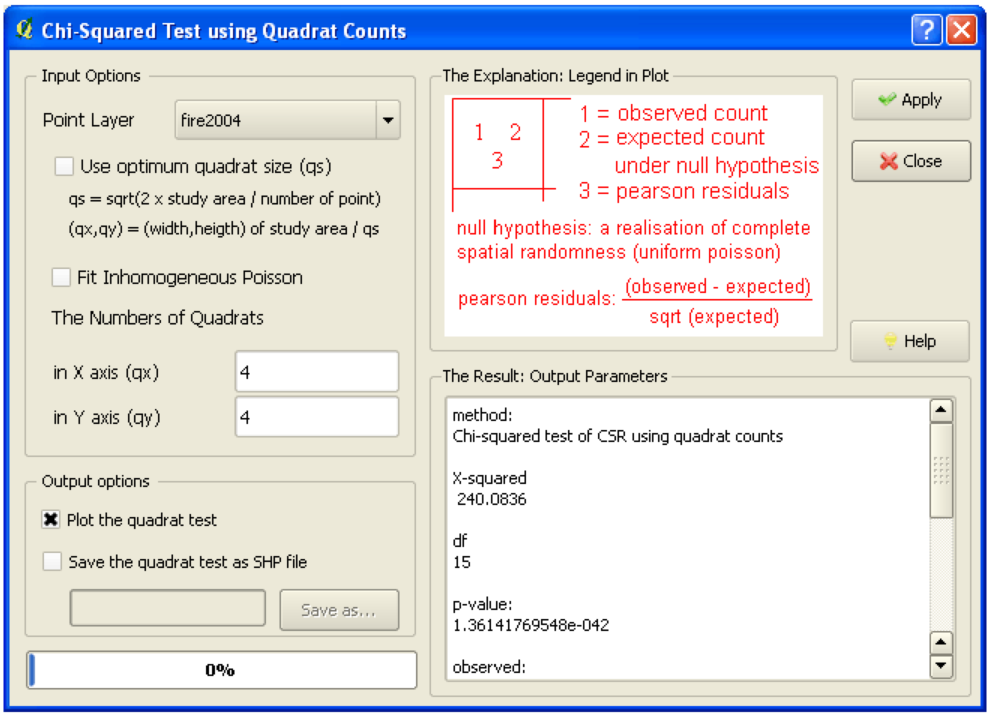
Plot only local function

For point id

Analysis Result: Local Version of Ripley's K (Left Side) and L (Right Side) Functions



7. Chi-Squared Test Using Quadrat Counts: GUI


Chi-Squared Test using Quadrat Counts

Input Options

Point Layer: fire2004

☐ Use optimum quadrat size (qs)

qs = sqrt(2 x study area / number of point)

(qx,qy) = (width,heigth) of study area / qs

☐ Fit Inhomogeneous Poisson

The Numbers of Quadrats

in X axis (qx): 4

in Y axis (qy): 4

Output options

☒ Plot the quadrat test

☐ Save the quadrat test as SHP file

0%

The Explanation: Legend in Plot

1 = observed count
2 = expected count
under null hypothesis
3 = pearson residuals

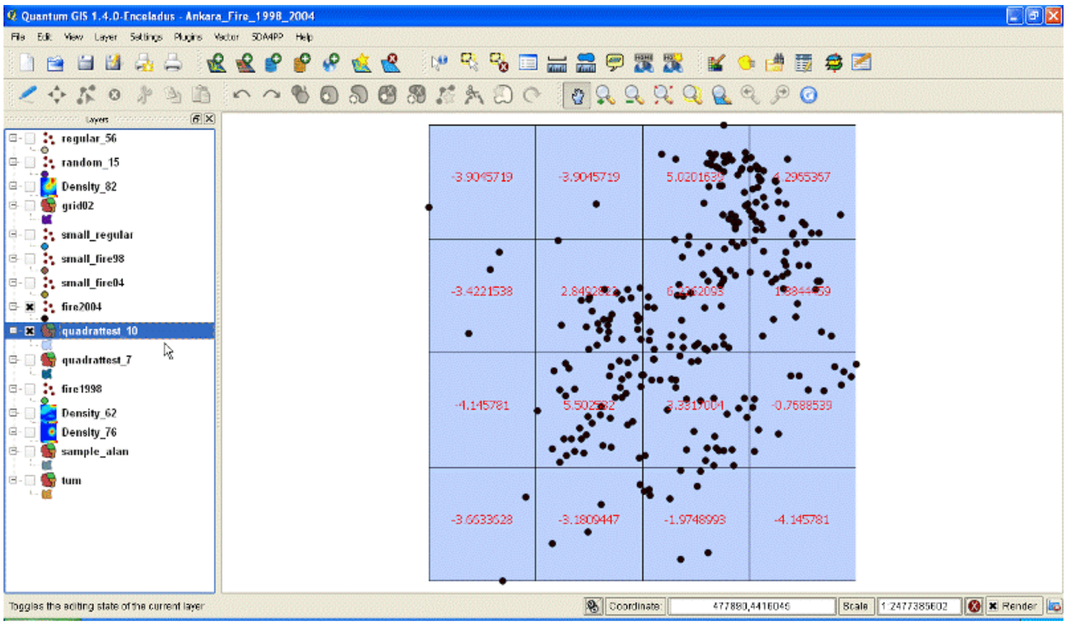
null hypothesis: a realisation of complete spatial randomness (uniform poisson)

pearson residuals: $\frac{\text{(observed - expected)}}{\text{sqrt (expected)}}$

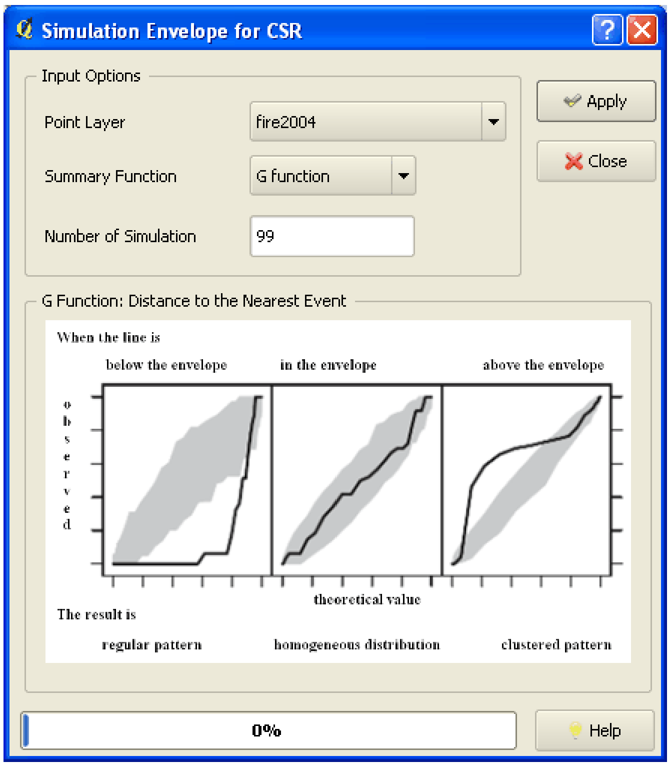
The Result: Output Parameters

```
method:
Chi-squared test of CSR using quadrat counts

X-squared
 240.0836

df
15

p-value:
1.36141769548e-042

observed:
```

Analysis Result



8. Simulation Envelope for CSR: GUI


Simulation Envelope for CSR

Input Options

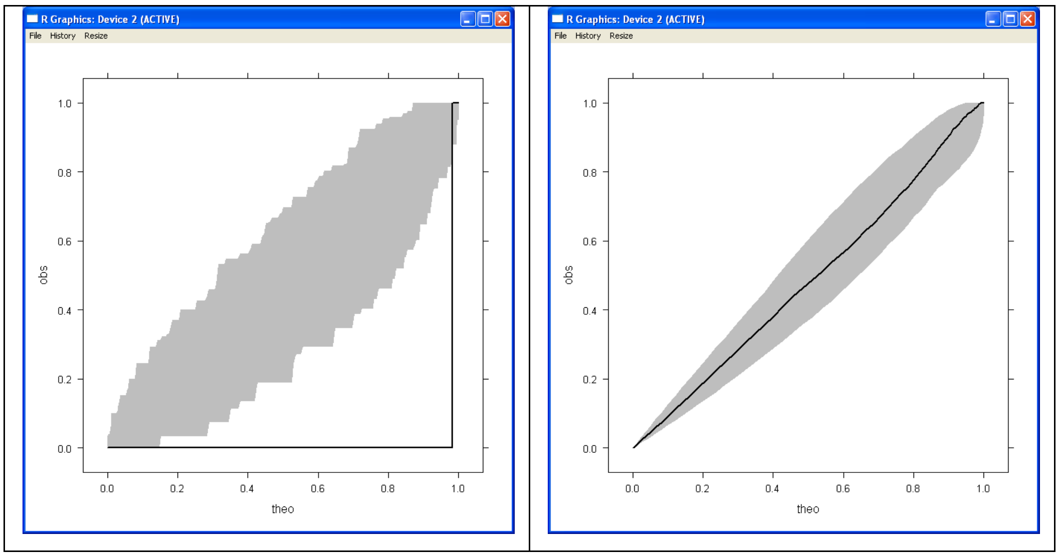
Point Layer: fire2004

Summary Function: G function

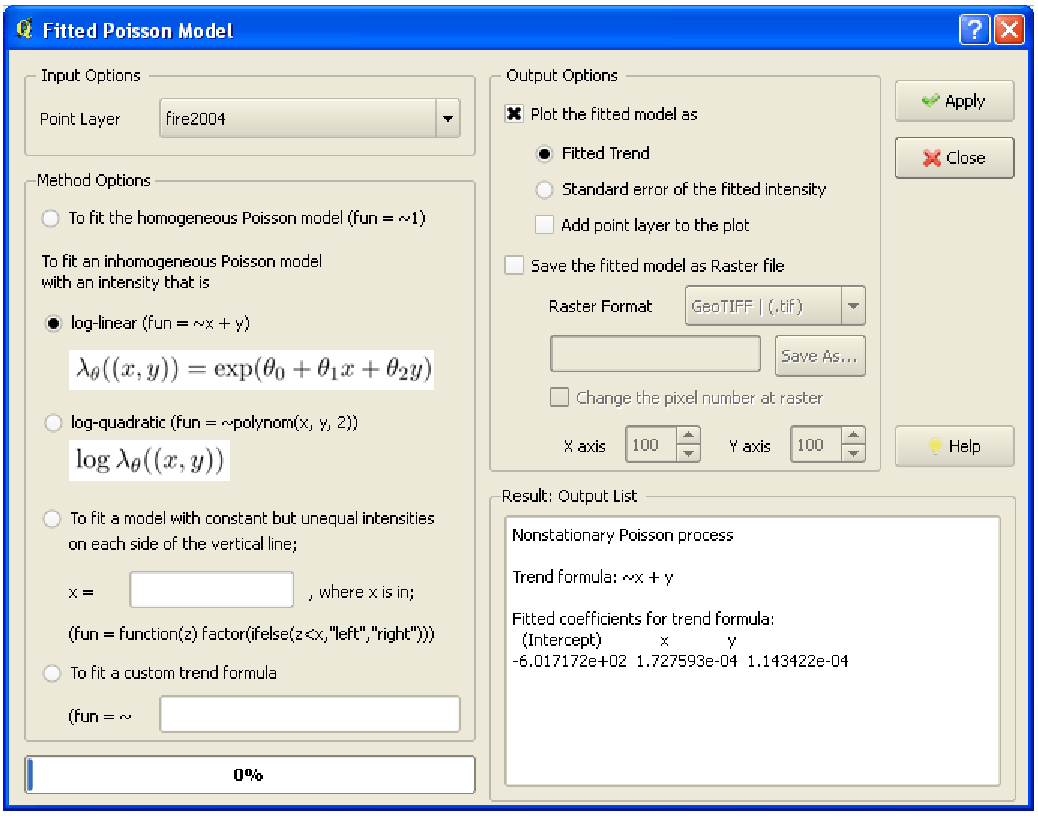
Number of Simulation: 99

G Function: Distance to the Nearest Event

Analysis Result: Simulation Envelope of G (Left Side) and F (Right Side) Function for CSR



9. Fitted Poisson Model: GUI



Input Options

Point Layer: fire2004

Method Options

( ) To fit the homogeneous Poisson model (fun = ~1)

To fit an inhomogeneous Poisson model with an intensity that is

(●) log-linear (fun = ~x + y)

$$\lambda_\theta((x,y)) = \exp(\theta_0 + \theta_1 x + \theta_2 y)$$

( ) log-quadratic (fun = ~polynom(x, y, 2))

$$\log \lambda_\theta((x,y))$$

( ) To fit a model with constant but unequal intensities on each side of the vertical line;

x = , where x is in;

(fun = function(z) factor(ifelse(z<x,"left","right")))

( ) To fit a custom trend formula

(fun = ~

0%

Output Options

[✓] Plot the fitted model as

(●) Fitted Trend

( ) Standard error of the fitted intensity

[ ] Add point layer to the plot

[ ] Save the fitted model as Raster file

Raster Format: GeoTIFF | (.tif)

Save As...

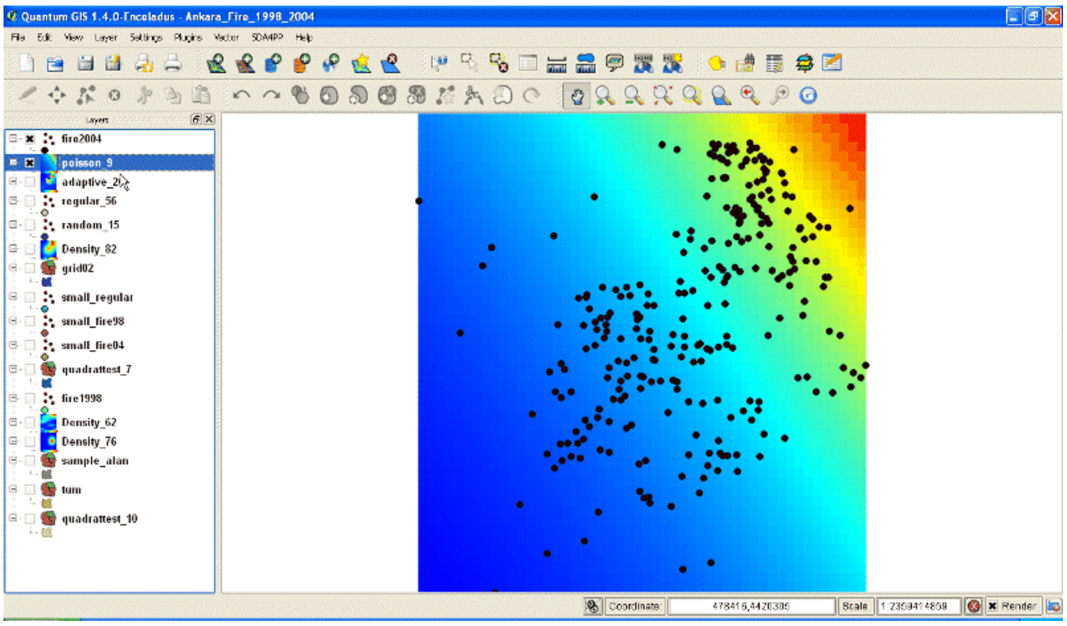
[ ] Change the pixel number at raster

X axis 100 Y axis 100

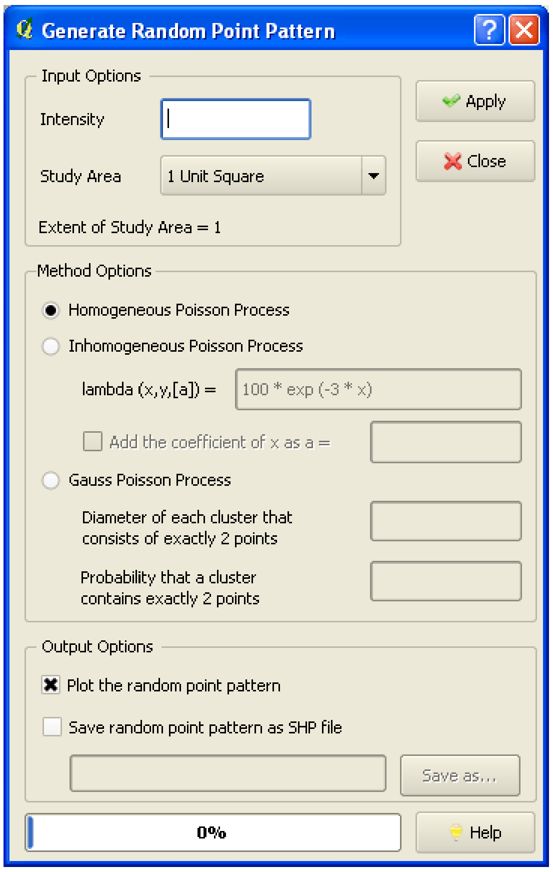
Result: Output List

```
Nonstationary Poisson process

Trend formula: ~x + y

Fitted coefficients for trend formula:
  (Intercept)           x            y
-6.017172e+02  1.727593e-04  1.143422e-04
```

Analysis Result



10. Generate Random Point Pattern: GUI



Generate Random Point Pattern

Input Options

Intensity:

Study Area: 1 Unit Square

Extent of Study Area = 1

Method Options

● Homogeneous Poisson Process

○ Inhomogeneous Poisson Process

lambda (x,y,[a]) = 100 * exp (-3 * x)

☐ Add the coefficient of x as a =

○ Gauss Poisson Process

Diameter of each cluster that consists of exactly 2 points

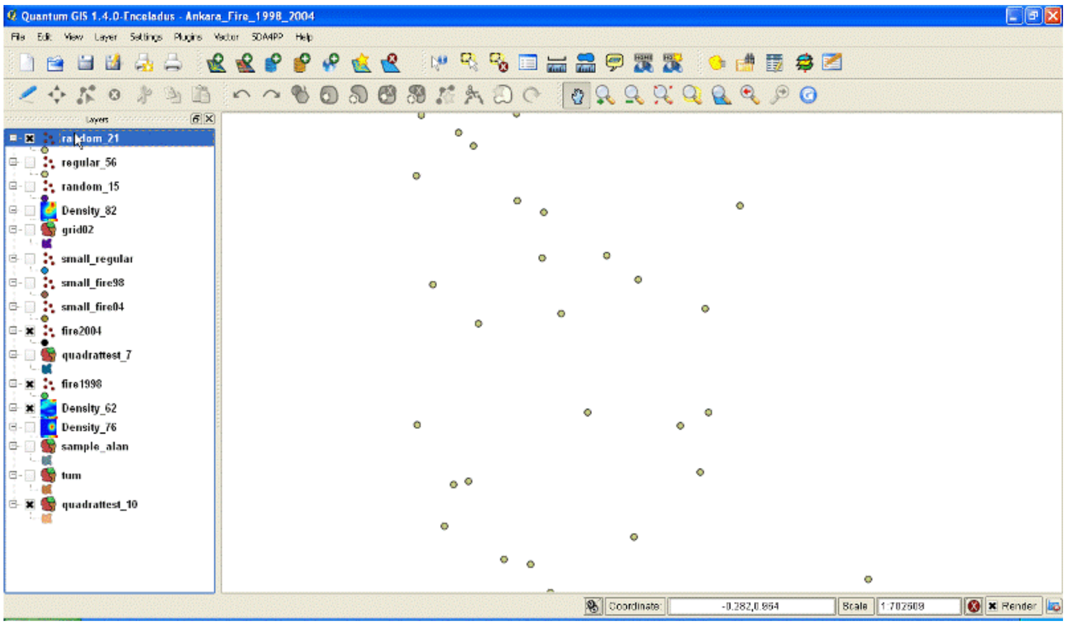
Probability that a cluster contains exactly 2 points

Output Options

☒ Plot the random point pattern

☐ Save random point pattern as SHP file

Result

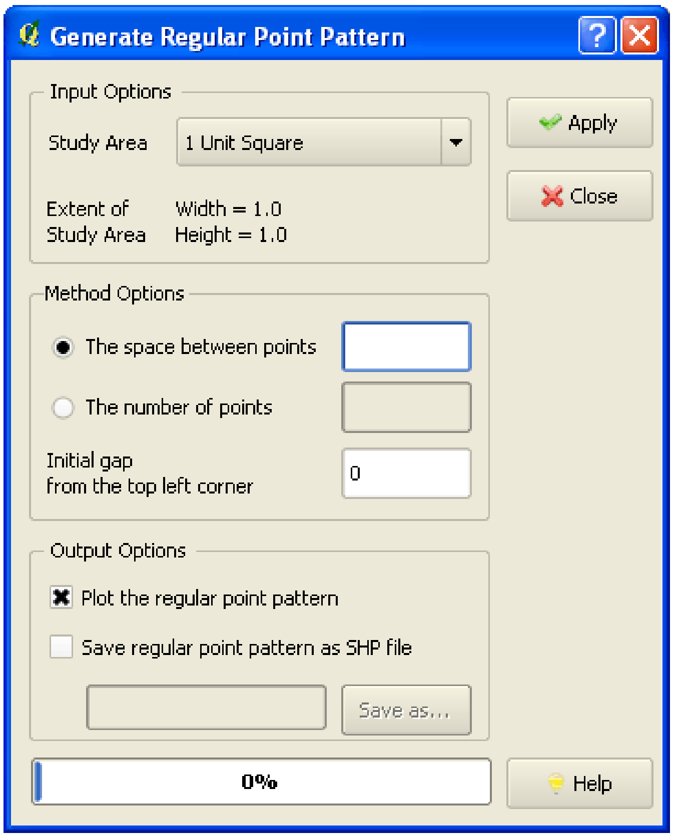


Quantum GIS 1.4.0-Enceladus - Ankara_Fire_1998_2004

Coordinate: -0.282,0.864

Scale 1:702609

11. Generate Regular Point Pattern: GUI



Generate Regular Point Pattern

Input Options

Study Area: 1 Unit Square

Extent of Study Area: Width = 1.0, Height = 1.0

Method Options

(●) The space between points

( ) The number of points

Initial gap from the top left corner: 0

Output Options

[✓] Plot the regular point pattern

[ ] Save regular point pattern as SHP file

Save as...

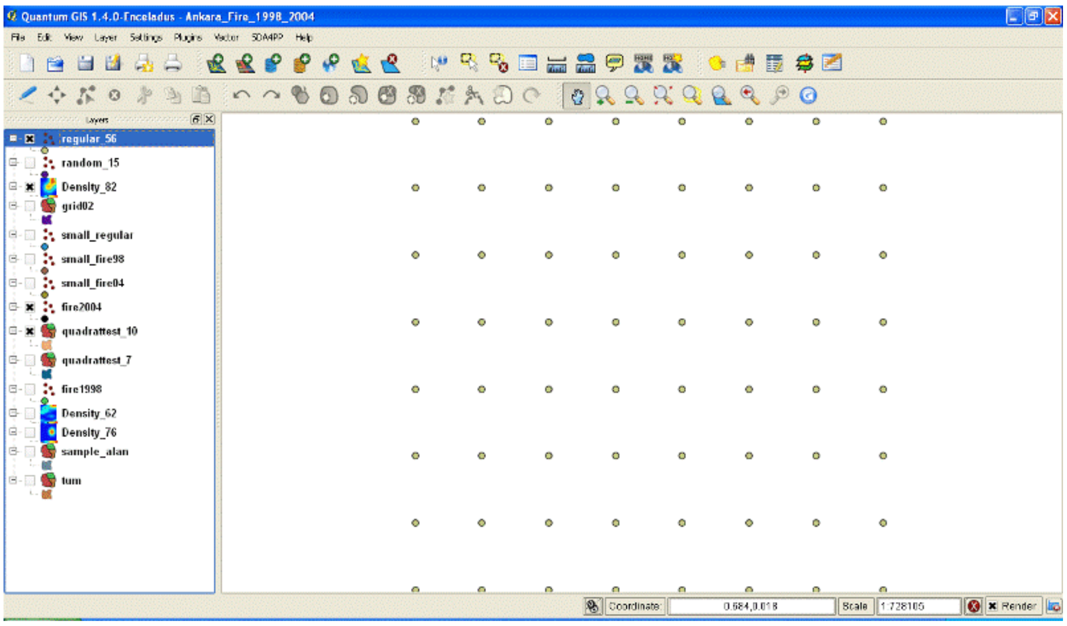
0%

Apply

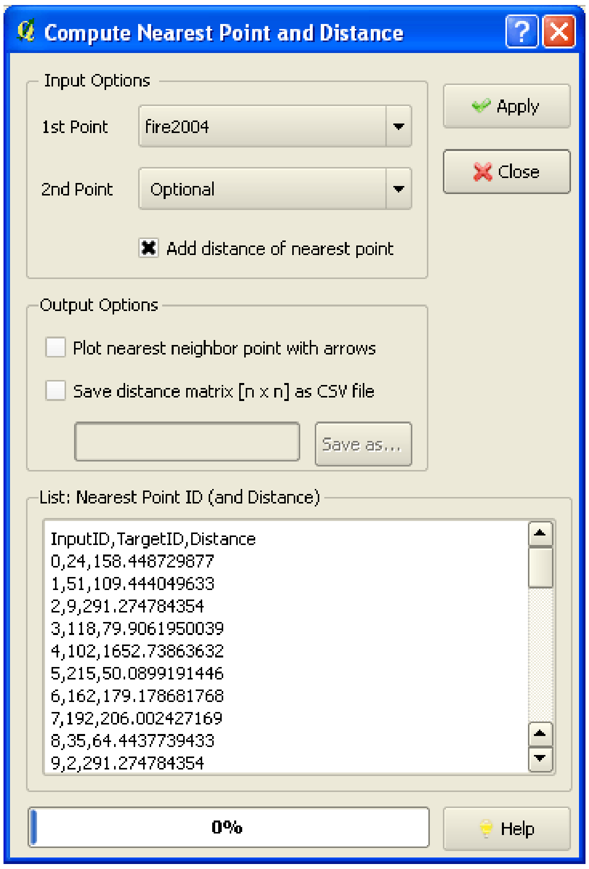
Close

Help

Result



12. Compute nearest Point and Distance: GUI



Input Options

1st Point: fire2004

2nd Point: Optional

[✓] Add distance of nearest point

Output Options

[ ] Plot nearest neighbor point with arrows

[ ] Save distance matrix [n x n] as CSV file

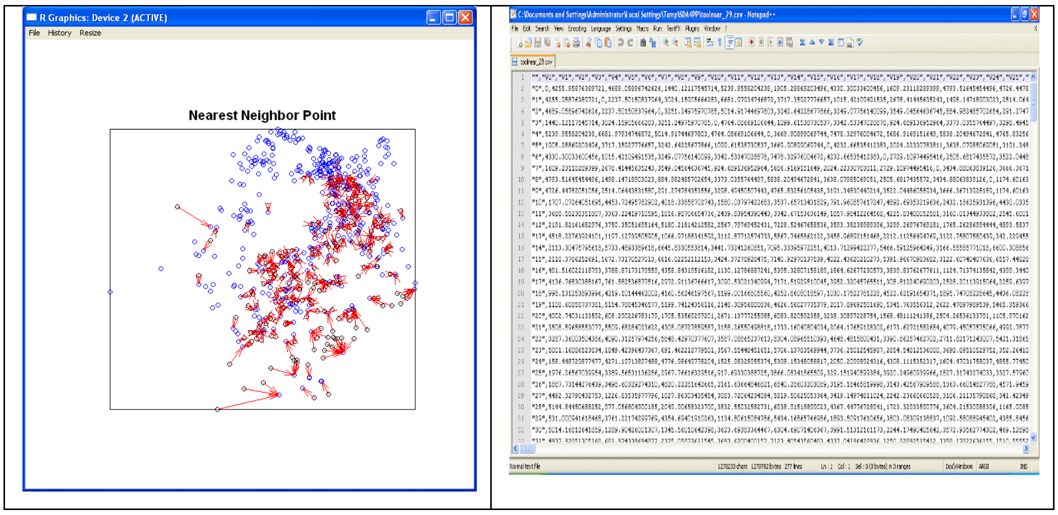
Save as...

List: Nearest Point ID (and Distance)

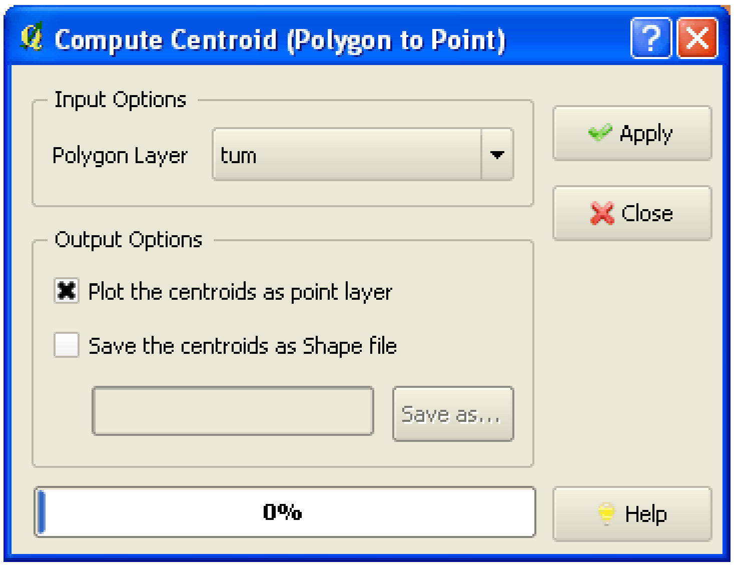
```
InputID,TargetID,Distance
0,24,158.448729877
1,51,109.444049633
2,9,291.274784354
3,118,79.9061950039
4,102,1652.73863632
5,215,50.0899191446
6,162,179.178681768
7,192,206.002427169
8,35,64.4437739433
9,2,291.274784354
```

0%

Result: Compute nearest Point (Left Side) and Distance (Right Side)



13. Compute Centroid (Polygon to Point): GUI


Compute Centroid (Polygon to Point)

Input Options

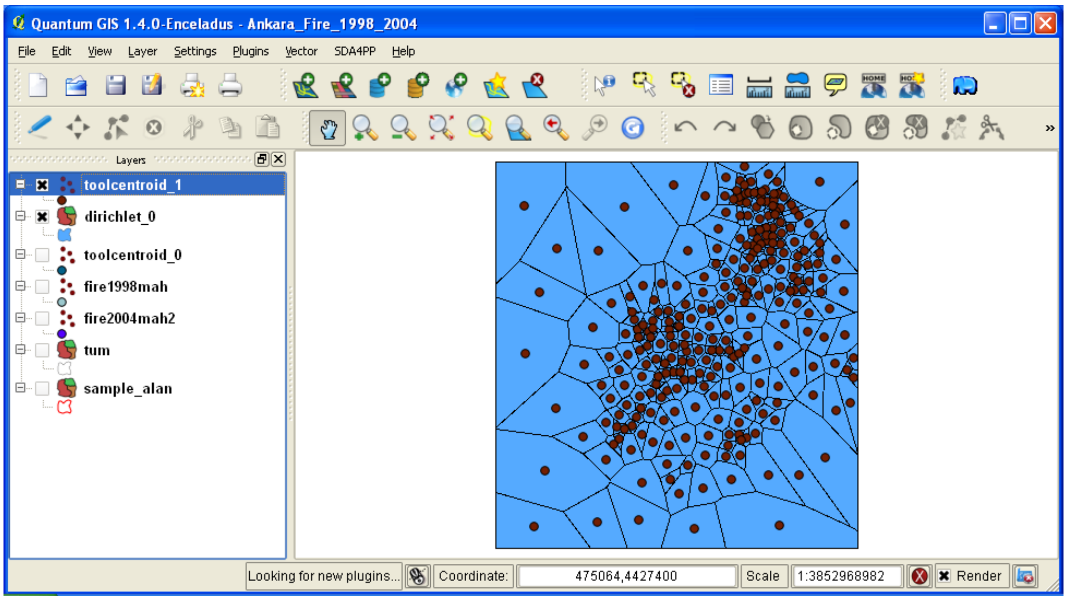
Polygon Layer: tum

Output Options

[✓] Plot the centroids as point layer

[ ] Save the centroids as Shape file

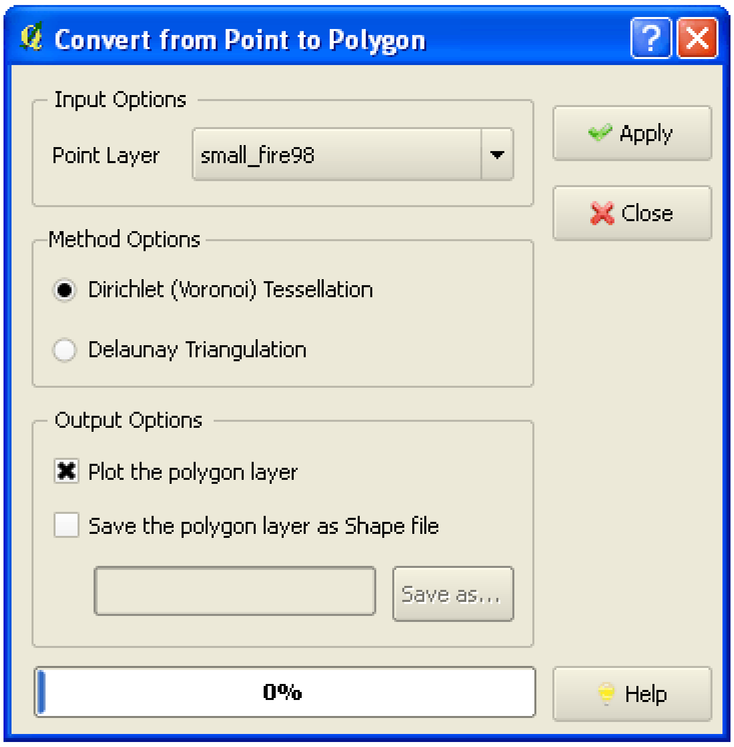
Result


Quantum GIS 1.4.0-Enceladus - Ankara_Fire_1998_2004

Layers:
- [✓] toolcentroid_1
- [✓] dirichlet_0
- [ ] toolcentroid_0
- [ ] fire1998mah
- [ ] fire2004mah2
- [ ] tum
- [ ] sample_alan

Looking for new plugins... Coordinate: 475064,4427400 Scale 1:3852968982 [✓] Render

14. Convert from Point to Polygon: GUI



Convert from Point to Polygon

Input Options

Point Layer: small_fire98

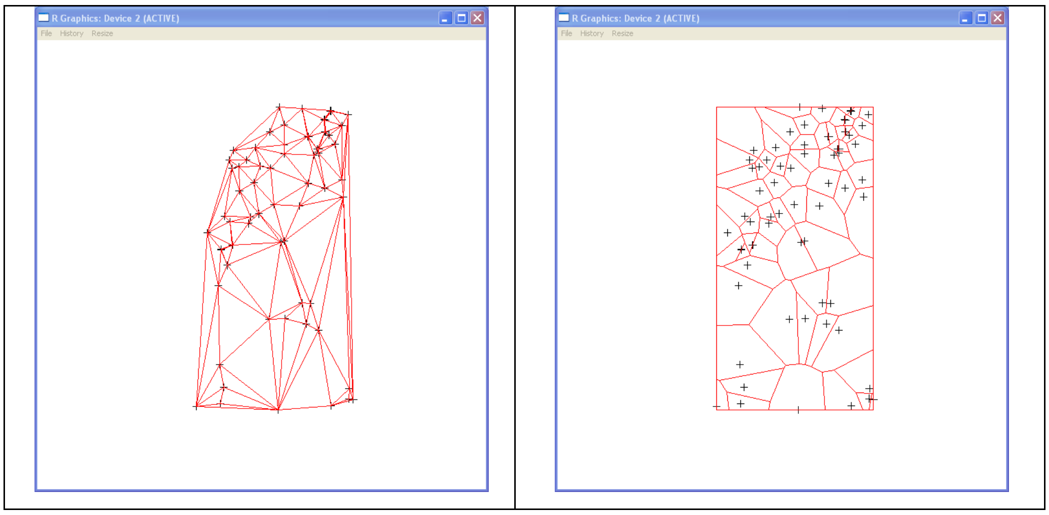
Method Options

- [✓ Dirichlet (Voronoi) Tessellation]
- Delaunay Triangulation

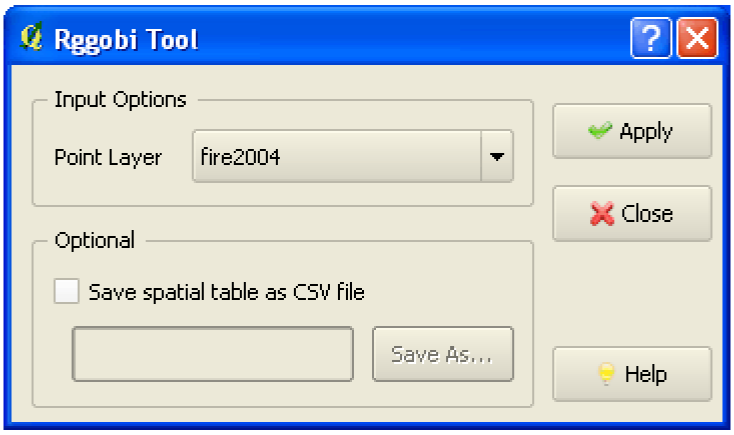
Output Options

- [✓ Plot the polygon layer]
- Save the polygon layer as Shape file

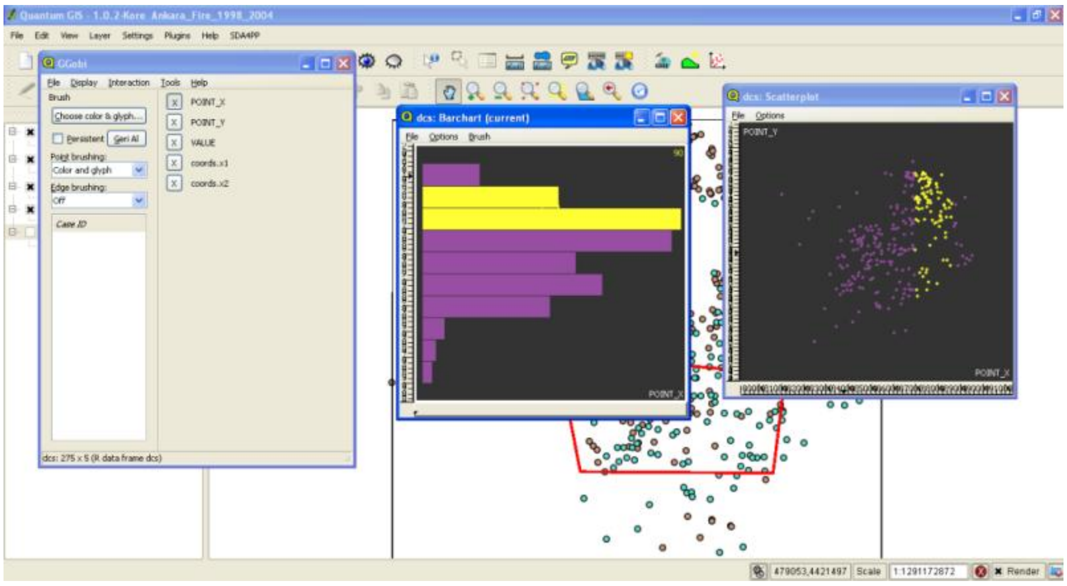
Result: Convert from Point to Polygon according to the Delaunay Triangulation (Left Side) and Dirichlet/Voronoi Tessellation (Right Side)



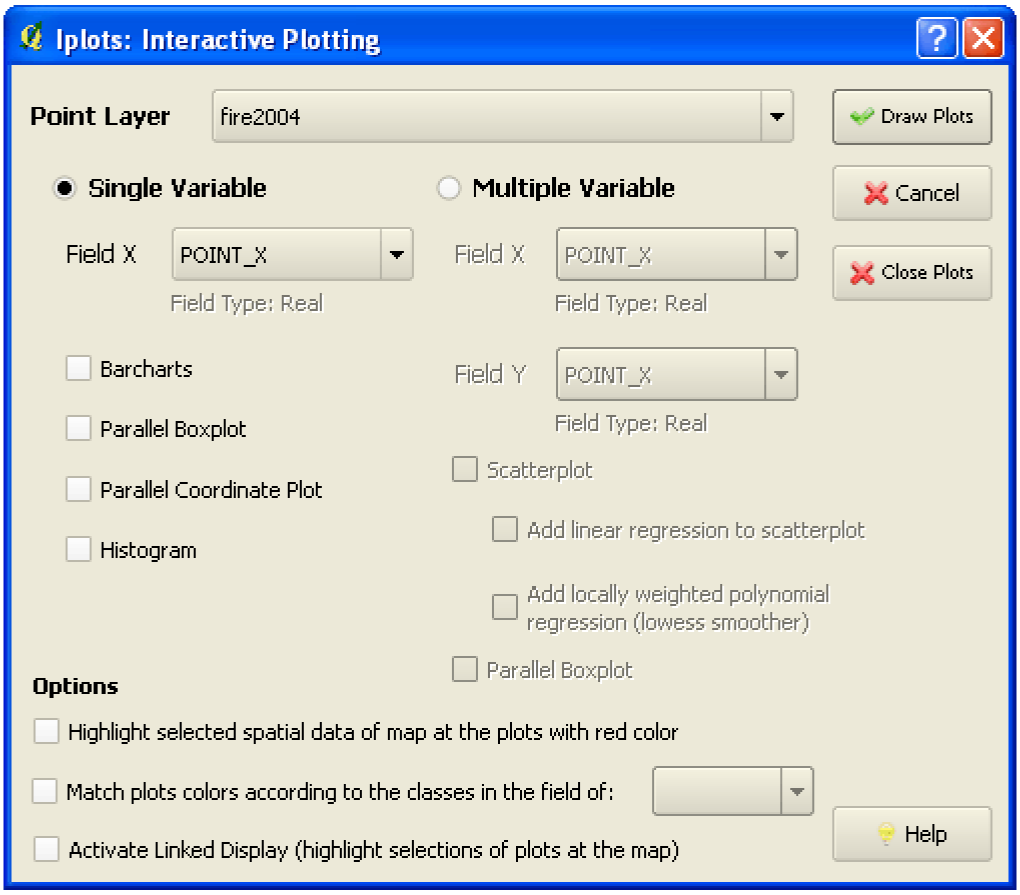
15. Rggobi Tool: GUI



Result



16. Iplots; Interactive Plotting: GUI



Iplots: Interactive Plotting

Point Layer: fire2004

(●) Single Variable ( ) Multiple Variable

Single Variable — Field X: POINT_X — Field Type: Real

- [ ] Barcharts
- [ ] Parallel Boxplot
- [ ] Parallel Coordinate Plot
- [ ] Histogram

Multiple Variable — Field X: POINT_X — Field Type: Real

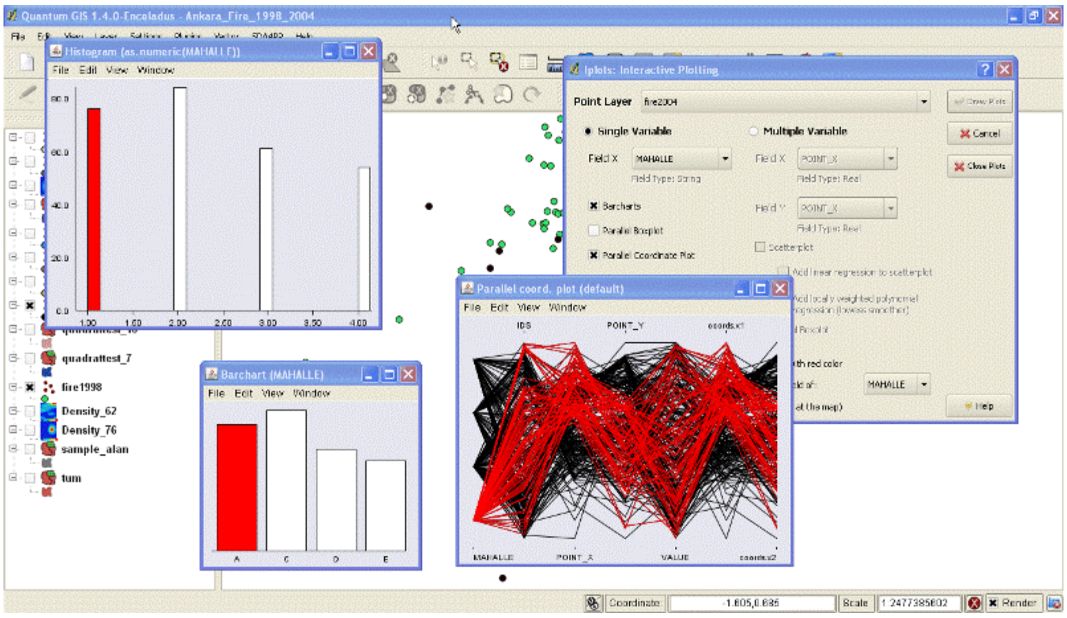
Field Y: POINT_X — Field Type: Real

- [ ] Scatterplot
- [ ] Add linear regression to scatterplot
- [ ] Add locally weighted polynomial regression (lowess smoother)
- [ ] Parallel Boxplot

Options

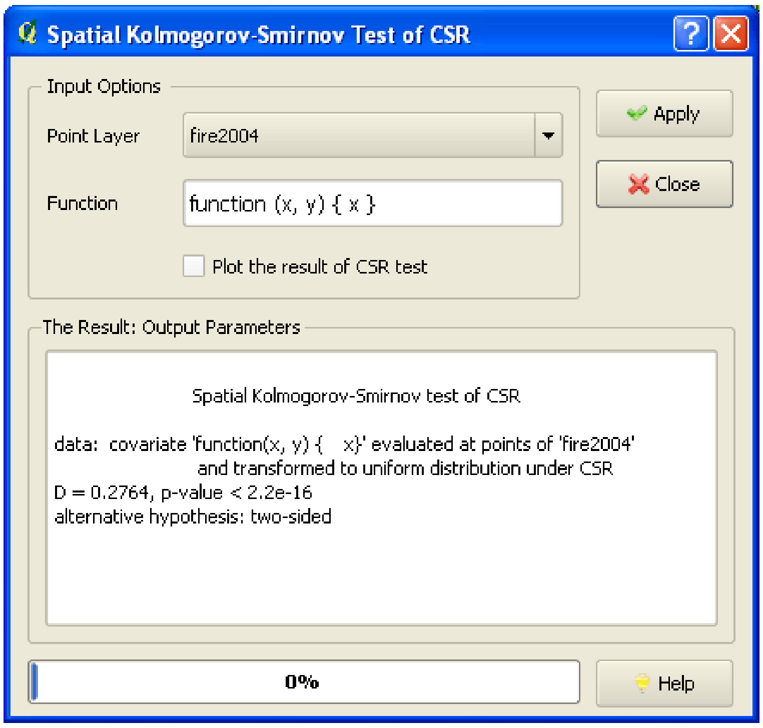
- [ ] Highlight selected spatial data of map at the plots with red color
- [ ] Match plots colors according to the classes in the field of:
- [ ] Activate Linked Display (highlight selections of plots at the map)

Result



Quantum GIS 1.4.0-Enceladus - Ankara_Fire_1998_2004
Histogram (as.numeric(MAHALLE))
Barchart (MAHALLE)
Parallel coord. plot (default)
Iplots: Interactive Plotting
quadrattest_7
fire1998
Density_62
Density_76
sample_alan
tum

17. Spatial Kolmogorov-Smirnov Test of CSR: GUI and Analysis Result


Spatial Kolmogorov-Smirnov Test of CSR
Input Options
Apply
Close
The Result: Output Parameters
0%
Help

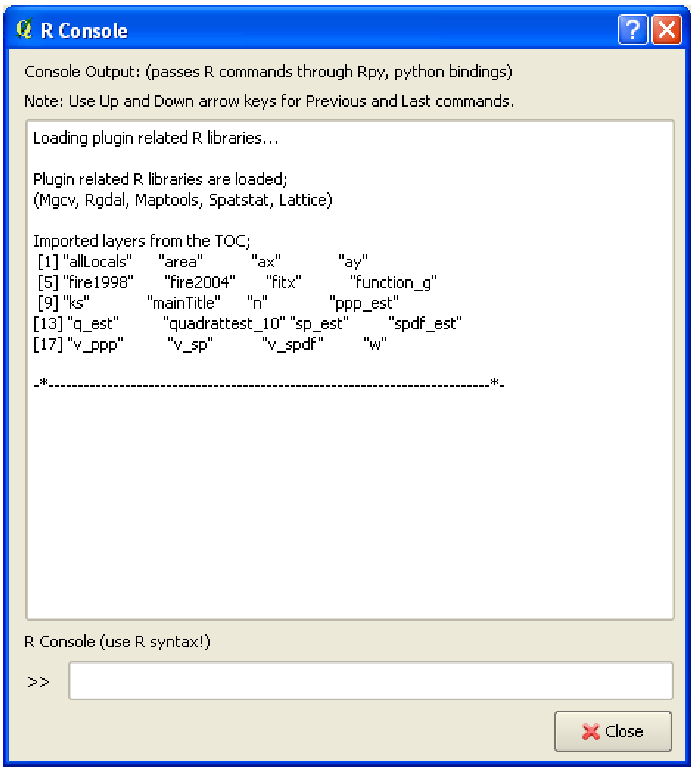
Point Layer: fire2004

Function: function (x, y) { x }

Plot the result of CSR test

```
                Spatial Kolmogorov-Smirnov test of CSR

data:  covariate 'function(x, y) {    x}' evaluated at points of 'fire2004'
                      and transformed to uniform distribution under CSR
D = 0.2764, p-value < 2.2e-16
alternative hypothesis: two-sided
```

18. R Console: GUI

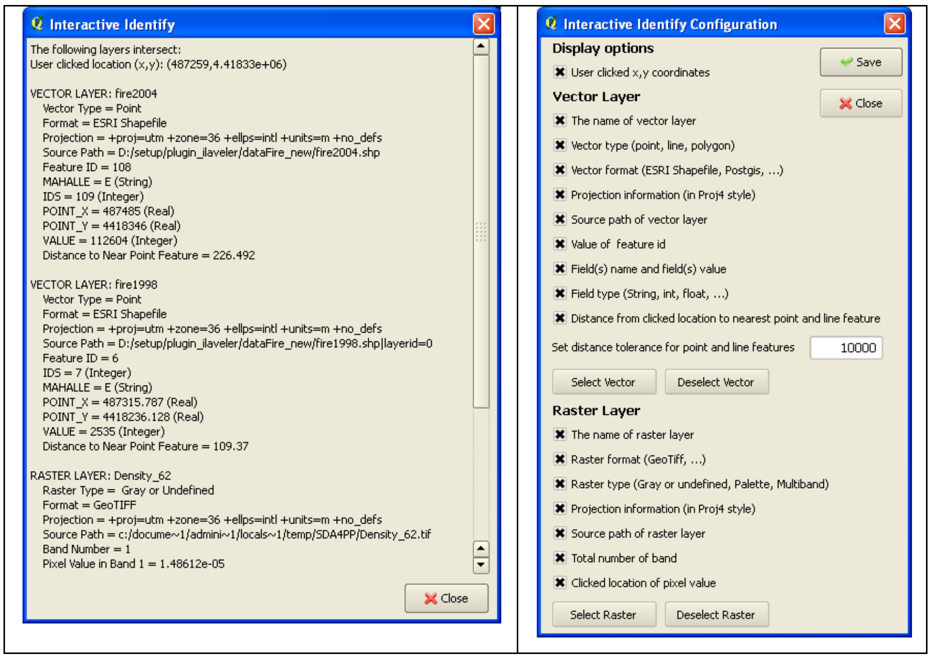


Console Output: (passes R commands through Rpy, python bindings)

Note: Use Up and Down arrow keys for Previous and Last commands.

```
Loading plugin related R libraries...

Plugin related R libraries are loaded;
(Mgcv, Rgdal, Maptools, Spatstat, Lattice)

Imported layers from the TOC;
 [1] "allLocals"     "area"         "ax"           "ay"
 [5] "fire1998"      "fire2004"     "fitx"         "function_g"
 [9] "ks"            "mainTitle"    "n"            "ppp_est"
[13] "q_est"         "quadrattest_10" "sp_est"     "spdf_est"
[17] "v_ppp"         "v_sp"         "v_spdf"       "w"

-*---------------------------------------------------------------------------*-
```

19. Interactive Identify Tool (Left Side) and Configuration (Right Side): GUI



Interactive Identify

```
The following layers intersect:
User clicked location (x,y): (487259,4.41833e+06)

VECTOR LAYER: fire2004
   Vector Type = Point
   Format = ESRI Shapefile
   Projection = +proj=utm +zone=36 +ellps=intl +units=m +no_defs
   Source Path = D:/setup/plugin_ilaveler/dataFire_new/fire2004.shp
   Feature ID = 108
   MAHALLE = E (String)
   IDS = 109 (Integer)
   POINT_X = 487485 (Real)
   POINT_Y = 4418346 (Real)
   VALUE = 112604 (Integer)
   Distance to Near Point Feature = 226.492

VECTOR LAYER: fire1998
   Vector Type = Point
   Format = ESRI Shapefile
   Projection = +proj=utm +zone=36 +ellps=intl +units=m +no_defs
   Source Path = D:/setup/plugin_ilaveler/dataFire_new/fire1998.shp|layerid=0
   Feature ID = 6
   IDS = 7 (Integer)
   MAHALLE = E (String)
   POINT_X = 487315.787 (Real)
   POINT_Y = 4418236.128 (Real)
   VALUE = 2535 (Integer)
   Distance to Near Point Feature = 109.37

RASTER LAYER: Density_62
   Raster Type =  Gray or Undefined
   Format = GeoTIFF
   Projection = +proj=utm +zone=36 +ellps=intl +units=m +no_defs
   Source Path = c:/docume~1/admini~1/locals~1/temp/SDA4PP/Density_62.tif
   Band Number = 1
   Pixel Value in Band 1 = 1.48612e-05
```

Interactive Identify Configuration

Display options

- [x] User clicked x,y coordinates

Vector Layer

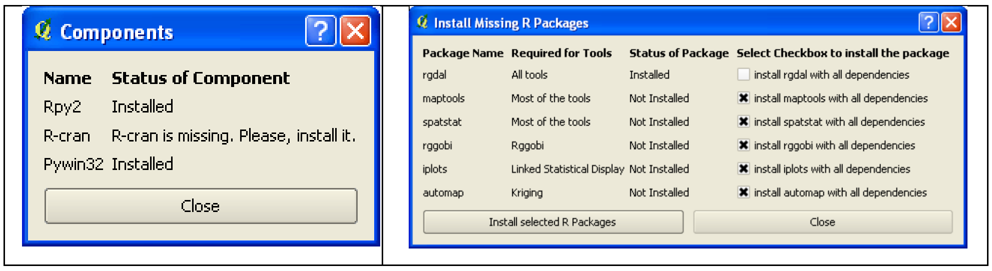
- [x] The name of vector layer
- [x] Vector type (point, line, polygon)
- [x] Vector format (ESRI Shapefile, Postgis, ...)
- [x] Projection information (in Proj4 style)
- [x] Source path of vector layer
- [x] Value of feature id
- [x] Field(s) name and field(s) value
- [x] Field type (String, int, float, ...)
- [x] Distance from clicked location to nearest point and line feature

Set distance tolerance for point and line features: 10000

Raster Layer

- [x] The name of raster layer
- [x] Raster format (GeoTiff, ...)
- [x] Raster type (Gray or undefined, Palette, Multiband)
- [x] Projection information (in Proj4 style)
- [x] Source path of raster layer
- [x] Total number of band
- [x] Clicked location of pixel value

20. Components (Left Side) and Install Missing R Packages (Right Side): GUI



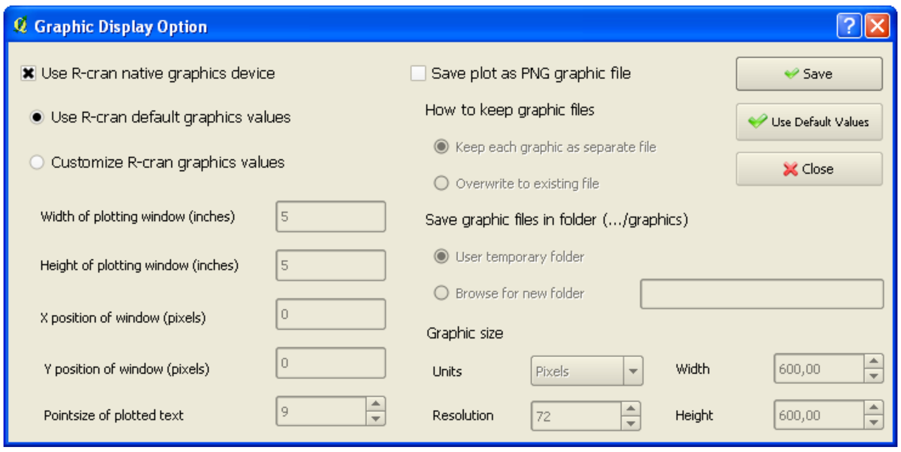
Components

Name	Status of Component
Rpy2	Installed
R-cran	R-cran is missing. Please, install it.
Pywin32	Installed

Install Missing R Packages

Package Name	Required for Tools	Status of Package	Select Checkbox to install the package
rgdal	All tools	Installed	☐ install rgdal with all dependencies
maptools	Most of the tools	Not Installed	☒ install maptools with all dependencies
spatstat	Most of the tools	Not Installed	☒ install spatstat with all dependencies
rggobi	Rggobi	Not Installed	☒ install rggobi with all dependencies
iplots	Linked Statistical Display	Not Installed	☒ install iplots with all dependencies
automap	Kriging	Not Installed	☒ install automap with all dependencies

21. Graphic Display Option: GUI



Graphic Display Option

- [x] Use R-cran native graphics device
  - [✓ Use R-cran default graphics values]
  - Customize R-cran graphics values

Width of plotting window (inches)	5
Height of plotting window (inches)	5
X position of window (pixels)	0
Y position of window (pixels)	0
Pointsize of plotted text	9

- [ ] Save plot as PNG graphic file

How to keep graphic files
- [✓ Keep each graphic as separate file]
- Overwrite to existing file

Save graphic files in folder (.../graphics)
- [✓ User temporary folder]
- Browse for new folder

Graphic size

Units	Pixels	Width	600,00
Resolution	72	Height	600,00

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name:	Kepoğlu, Volkan Osman
Nationality:	Turkish
Date and Place of Birth:	8 January 1977, Ankara
Marital Status:	Married, One Child
Phone:	+90 532 705 65 71
E-mail:	vkepoglu@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
MS	METU, Urban Design	2003
BS	METU, City and Regional Planning	2000
High School	Gazi Anatolian High School, Ankara	1995

WORK EXPERIENCE

Year	Place	Enrollment
2009 – 2010	NABUCCO Natural Gas Pipeline Project	Technical GIS Advisor
2004 – 2005	NABUCCO Natural Gas Pipeline Project	Technical GIS Advisor
2003 – 2011	BTC Crude Oil Pipeline Project	Mapping & GIS Manager
2001 – 2003	BTC Crude Oil Pipeline Project	GIS Expert
2001 – 2001	Returning to Village & Rehabilitation Proj.	City Planner
2000 – 2001	METU Cyprus Campus Design Project	Urban Designer
2000 – 2000	Gallipoli Historical National Park Project	Research Assistant

FOREIGN LANGUAGES

Advanced English, Beginner German

PUBLICATIONS

1. Kepoğlu, V. O., and Düzgün, S., (2008), "Development of Free GIS Software: The Example of Quantum GIS", 2nd Remote Sensing and Geographic Information System Symposium 2008, Kayseri, Turkey, October, 13-15.

2. Kepoğlu, V. O., and Usul, N., (2005), "Representation of Socio Economic Indicators on Health Status in Turkey", ESRI International User Conference 2005, San Diego, USA, July, 25-29.

HOBBIES

Swimming, Football, Chess, Movies, Bowling, Table Tennis